PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR

FACULTAD DE INGENIERÍA

ESCUELA DE SISTEMAS

DESARROLLO DE UN JUEGO INTERACTIVO CON UNA CÁMARA MICROSOFT KINECT PARA EL RECONOCIMIENTO DEL JUGADOR Y LOS DIFERENTES CORTES DE ESPADA.

ALUMNO:
DAVID ALEJANDRO ZURITA ALTAMIRANO

QUITO 2013

AGRADECIMIENTOS

El agradecimiento más profundo de mi alma a mi madre, que siempre estuvo ahí para mí y que me ha apoyado durante todo este tiempo; por ser la mejor madre y el mejor maestro.

DEDICATORIA

Este trabajo lo dedico a mi padre, sabiendo que me hubiese apoyado en cada paso de este camino, y más que nada siguiendo su consejo: “Sky is the limit”.

RESUMEN

La presente disertación de grado, presenta el diseño y desarrollo de un juego en Processing con una cámara Microsoft Kinect, para la interpretación de movimientos del usuario para realizar cortes de espada.

Se empezó con un leve criterio de la posibilidad de explicar de diferente modo los cortes enseñados en el arte marcial Bujinkan, fuera del dojo en donde se pueda tener una idea de las bases de los cortes.

Esta idea concebida se desarrolla en el presente documento, en donde se adaptó la tecnología para presentar un juego básico que reconoce el espacio frente al usuario y los cortes básicos que se enseñan.

ÍNDICE

Tabla de Contenidos

ÍNDICE FIGURAS

ÍNDICE TABLAS

Capítulo I Marco Teórico

El presente capitulo explicará las tecnologías que se ha seleccionado para la presente disertación de grado. Cada una de estas presenta sus propias cualidades, funcionalidades y las razones del porqué han sido seleccionadas para el desarrollo de la presente disertación de grado.

1.1. Cámaras de análisis de profundidad.

Para comprender el funcionamiento de las cámaras de análisis de profundidad, primero se debe entender cómo funcionan, por ello se empieza explicando la radiación infrarroja.

1.1.1. Radiación Infrarroja:

La radiación infrarroja fue descubierta por el astrónomo Sir Frederick William Herschel, nacido en Alemania el 15 de noviembre de 1738, fallecido el 25 de agosto de 1822; su descubrimiento se dio mientras realizaba experimentos con luz, cuando acerco un termómetro de mercurio al espectro obtenido por un prisma de cristal al refractar la luz[1], obteniendo medidas de más calor cuando había acercado el termómetro pasando el rojo visible, llamó a este fenómeno “rayos caloricos”, no fue hasta pasado el siglo que se lo conoció como radiación infrarroja.

Esta radiación es imperceptible para el ojo humano, ya que es una radiación electromagnética de la luz con longitud de ondas más largas que las visibles, estas ondas se extienden del color rojo visible de luz, siendo estas ondas de una longitud de 700 nm a 1mm; con una frecuencia de onda de 430 THz bajando hasta 300 GHz.

[1] “Radiación Infrarroja”, Wikipedia Org. Última vez modificado 28 Oct. 2013, http://es.wikipedia.org/wiki/Radiaci%C3%B3n_infrarroja

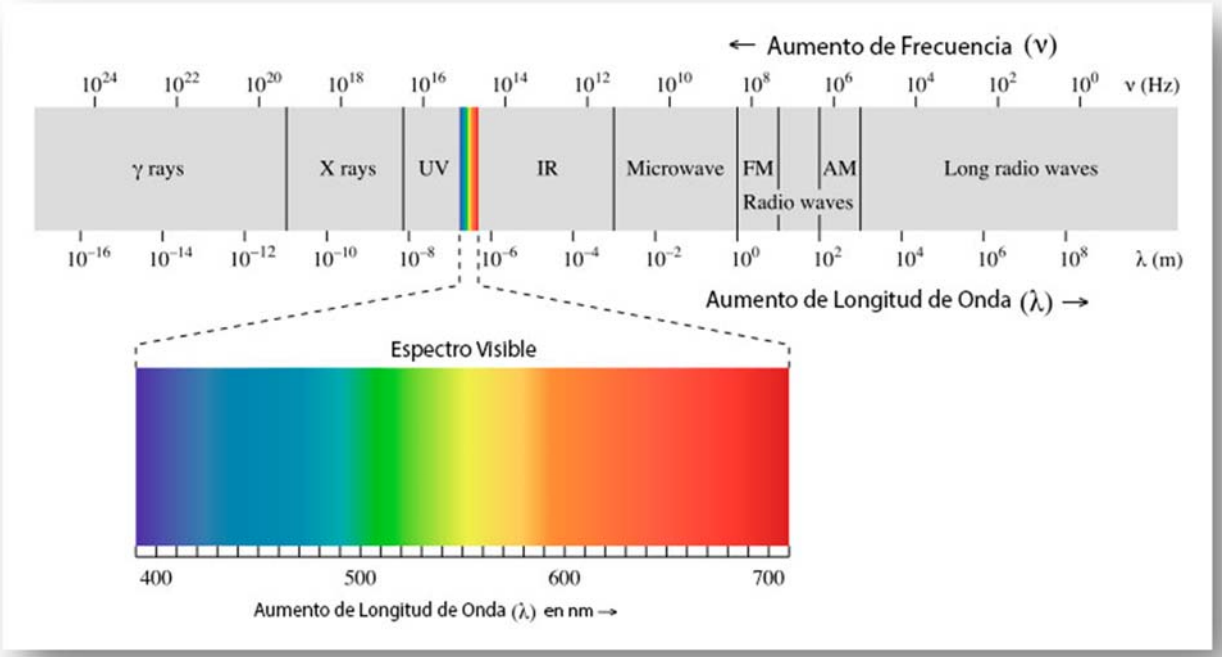

Figura 1- Descripción de ondas visibles y no visibles de la luz
Fuente: Wikipedia, Light Spectrum[2]
Traducido por David Zurita.

La figura 1-1 identifica el espectro visible, siendo éste todo lo que el ojo humano puede ver, abajo del color rojo encontramos la radiación infrarroja o IR, por sus siglas en inglés; dentro de esta encontramos una subdivisión de regiones que se ha creado para sus diferentes usos.

[2] "Light", Wikipedia Org. Última vez modificado 2 Nov. 2013, http://en.wikipedia.org/wiki/File:EM_spectrum.svg

Nombre de División	Abreviatura	Longitud de Onda	Características
Infrarrojo Cercano	NIR, IR-A DIN	0.75–1.4 μm	Uso común en las telecomunicaciones de fibra óptica, también para los dispositivos de visión nocturna.
Longitud de Onda Corta-IR	SWIR, IR-B DIN	1.4-3 μm	Región utilizada para telecomunicaciones de larga distancia.
Longitud de Onda Media-IR	MWIR, IR-C DIN; MidIR, IIR	3–8 μm	Con esta onda se puede detectar solo temperaturas mayores al calor del cuerpo, tecnología utilizada por misiles rastreadores.
Longitud de Onda Larga-IR	LWIR, IR-C DIN	8–15 μm	Imágenes Térmicas, estas ondas son utilizadas para detectar objetos con calor dentro de un cuarto.
Infrarrojo Lejano	FIR	15–1,000 μm	Se utiliza para detectar explosivos y armas químicas.

Tabla 1-: Cuadro de divisiones de Radiación Infrarroja y sus características.
Fuente: Wikipedia, Light Spectrum[3]
Elaborado y traducido por: David Zurita.

Se puede apreciar en la tabla 1-1 los rangos de las ondas IR y como estas son utilizadas para diferentes tecnologías. Se ha desarrollado un sin número de aplicaciones para las diferentes longitudes de onda, y estas han traído nuevas ideas para que tecnologías puedan ser posibles, así como ser aplicadas en diferentes campos. Para enumerar un poco de estas tecnologías:

- Visión Nocturna.
- Termografía.
- Fotografía Infrarroja.
- Filtros Infrarrojos.
- Cámaras Térmicas.
- Comunicaciones.
- Espectroscopia.
- Meteorología.
- Climatología.
- Astronomía.
- Imágenes Hiperespectrales

[3] "Infrared", Wikipedia Org. Última vez modificado 9 Nov. 2013,http://en.wikipedia.org/wiki/Infrared

Este último, las imágenes hiperespectrales, son las que se tomarán en cuenta para la cámara infrarroja dentro del Microsoft Kinect que se utilizará en este trabajo, ya que estas imágenes son las utilizadas dentro de las cámaras de vigilancia, son capaces de dividir la luz que capturan en más de tres capas, siendo así posible la visión nocturna con profundidad. Este último es el punto más importante de todos, ya que al ser capaz la cámara de percibir la profundidad de la imagen genera las posibilidades que explota el sistema creado por Microsoft para el Kinect.

Funcionamiento de las imágenes: Al adquirir la luz dentro del lente para la imagen, esta no solo la divide en la luz visible, también recolecta el espectro electromagnético de los objetos siendo capturados, esto produce una serie de imágenes capturadas por cada espectro electromagnético capturado, las imágenes son procesadas y colocadas en una sola imagen, esta diferenciación de espectros recolectados es lo que se percibe como la profundidad, esta tecnología es altamente usada para la agricultura, minería; ya que al tomar fotos satelitales de los espacios a ser explotados y utilizar los diferentes espectros recolectados para la búsqueda de diferentes residuos en la tierra, con el fin de buscar una mejor manera de como explotar este terreno.

1.1.2. Cámaras Infrarrojas

Una vez comprendido que tecnología se utiliza dentro de las cámaras infrarrojas, se explica cómo éstas recolectan la información para presentarla, e ir enfocando en la cámara específica dentro del Microsoft Kinect.

Para esto se expone el funcionamiento de la cámara dentro del Microsoft Kinect. El sensor de pixeles activos o APS, por sus siglas en inglés, es un sensor de imágenes que contiene un arreglo de sensores de pixeles, cada pixel contiene un foto detector y un amplificador activo.[4] A este sensor se lo conoce como un sensor CMOS, por sus componente CMOS, semiconductor complementario de metal-oxido, este semiconductor construido por circuitos integrados y desarrollado por Frank Wanlass en 1963[5], son los que se encargan de receptar la luz que entra por el lente de la cámara, para que los fotos detectores que están compuestos de los circuitos CMOS APS sean capaces de almacenar la luz que pasa en los pixeles, siendo esta luz tanto la luz perceptible por el ojo humano y ondas infrarrojas, el chip APS se encarga de juntar toda la información receptada por estos

[4] "Active pixel sensor", Wikipedia Org. Última vez modificado 28 Sep. 2013, http://en.wikipedia.org/wiki/Active_pixel_sensor
[5] "CMOS", Wikipedia Org. Última vez modificado 12 Nov. 2013, http://en.wikipedia.org/wiki/CMOS

fotos detectores y presentar los arreglos de pixeles en una forma que se le especifique, para nuestro caso es en tonos de grises, esto se explicará más adelante ya cuando se trate del Microsoft Kinect, esta imagen presentada no solo muestra la forma y colores que tienen los objetos al ser capturados, sino también la distancia que estos están con relación a la cámara.

1.2. Microsoft Kinect.

Proyecto “Natal”, proyecto de Microsoft con PrimeSense, empresa Israelí creadora del chip capaz de crear video con profundidad, ellos son los creadores del proyecto, con la visión de cambiar el mundo de los video juegos; dicho proyecto fue presentado en Norte América el 4 de noviembre del 2010, con el nombre de Kinect, cuando salió al mercado vendió más de 10 millones de unidades en el primer mes, marcando el record de venta más rápida de dispositivo electrónico[6]. Kinect fue desarrollado para eliminar el control de las consolas de video juegos y usar al usuario como dicho control para crear los comandos dentro de un video juego con gestos o sonidos; este fue un gran paso tecnológico, ya que nunca antes se había logrado algo así. Este dispositivo trae consigo un proyector infrarrojo, una cámara RGB, un sensor de profundidad, micrófonos localizados alrededor del dispositivo y un motor localizado en la base.

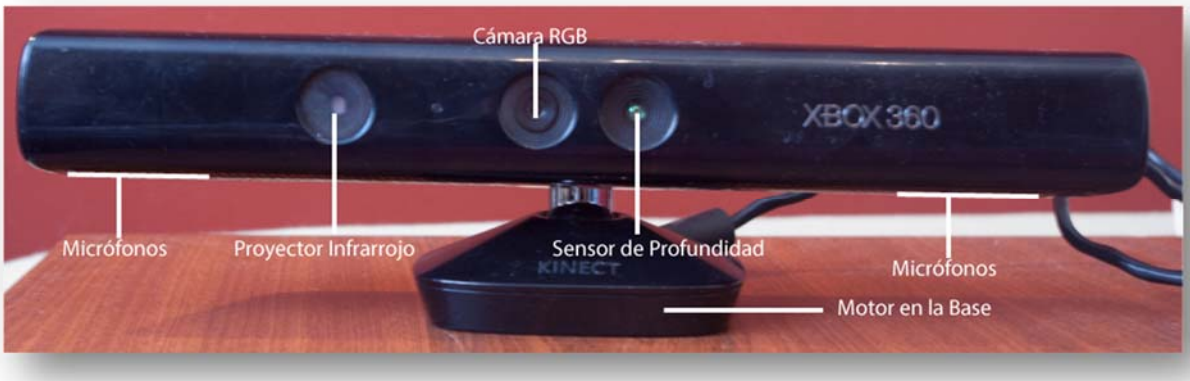

Figura 1-: Microsoft Kinect y sus partes
Elaborado por: David Zurita

[6] GIORIO, Clemente; FASCINARI, Massimo. Kinect in Motion – Audio and Visual Tracking by Example, Reino Unido: Packt Publishing, Abril 2013 1ra edición. 112 p.

1.2.1. Cámara RGB:

La cámara RGB está localizada en la mitad del dispositivo, como se podía apreciar en la figura 1-2, esta es una cámara web común, ya que su función es simplemente el de captar los objetos frente a ella como lo hace el ojo humano, con los colores respectivos. Esta cámara posee una resolución de 1280 x 960 pixeles con 12 imágenes por segundo; se puede llegar a tener 30 imágenes por segundo o frames per second (fps), pero con una reducción de resolución hasta de 640 x 480.

1.2.2. Proyector Infrarrojo y Sensor de Profundidad:

Estos dos trabajan conjuntamente para poder crear las imágenes de profundidad.

El proyector infrarrojo proyecta una cuadricula de puntos invisibles para el ojo humano, la función de esta cuadricula es identificar la distancia en la que se encuentran los objetos respecto al proyector, ahí es donde entra el sensor de profundidad, ya que este es el receptor de la cuadricula de puntos; al proyectar los puntos, estos rebotan al chocar contra algo, estos regresan hacia el sensor de profundidad del Kinect, este captura los puntos y los analiza, siendo así capaz de comprender la distancia en que se encuentran las cosas, y generar una visualización de los objetos que se encuentran en frente a la cámara, esto no solo muestra la localización de dichos objetos, también el análisis de la cuadricula determina la distancia que se encuentran los objetos en relación con el Kinect; esto es lo que permite que la captura y reconocimiento del “esqueleto” del usuario que se encuentra en frente a la cámara, de aquí nace la idea de que el usuario sea el control, y con sus movimientos este sea capaz de controlar el juego. El esqueleto que identifica el Kinect, son las articulaciones básicas del usuario interpretadas, la cámara Kinect tiene la capacidad de interpretar desde 1 esqueleto hasta 6, dos usuarios los completamente reconocibles, con sus articulaciones independientes, los 4 restantes son interpretables pero no son exactos, por ello solo se han visto aplicaciones con dos usuarios dentro de un juego que maneja un Kinect.

La cámara da la capacidad de trabajar con profundidad de imágenes, esto quiere decir que trabaja con tres ejes de coordenadas (x, y, z), y tenemos una resolución máxima del sensor de profundidad de 640 x 480 pixeles, tal vez no sea la mayor resolución, pero esto no es importante ya que cada pixel que contiene esta resolución tiene base 11 bits, lo que quiere decir que cada pixel es capaz de almacenar 2048 niveles de profundidad[7], esta

[7] Id.

cantidad de información representa muchas variaciones de distancia, lo cual permite una representación muy exacta de la distancia de los objetos, a esto debemos tener en cuenta el rango máximo de la proyección de la cuadricula la cual va desde 0.8 m a 4 m como una distancia mínima y máxima, respectivamente.

Figura 1-: Cuadricula de puntos Infrarrojos
Fuente: Making Things See: 3D vision with Kinect, Processing, Arduino, and MakerBot
Elaborado por: BORENSTEIN, Greg[8]

Esta cuadricula de puntos es lo que permite que se visualice la profundidad, para ver esto se ha realizado una captura de pantalla.

[8] BORENSTEIN, Greg. Making Things See: 3D vision with Kinect, Processing, Arduino, and MakerBot. Estados Unidos de Norte America: O'Reilly**,** Febrero 2012, 1ra edición. 440 p

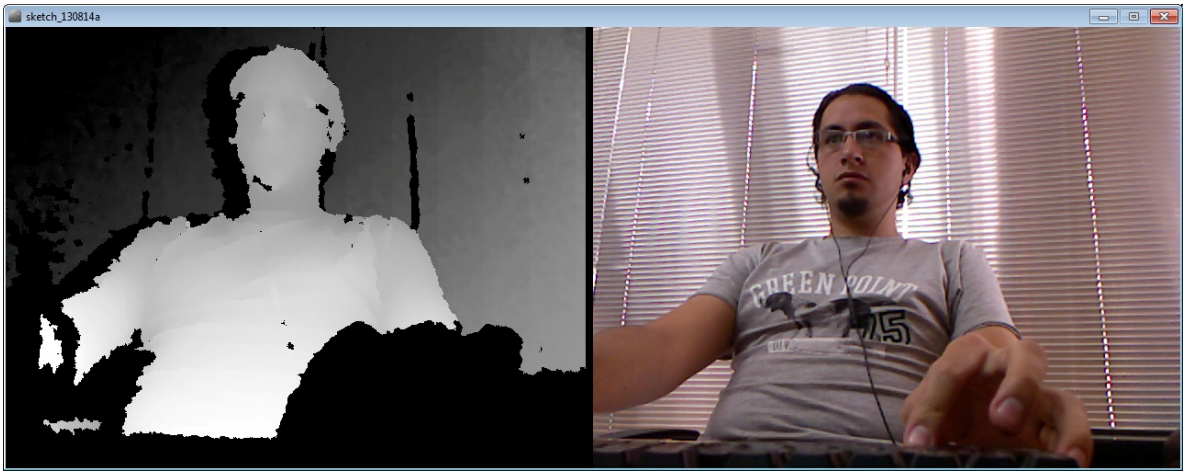

Figura 1-: Imagen de Profundidad vs Imagen a Color.
Fuente: Captura de pantalla
Elaborado por: David Zurita

En la figura 1-4 podemos ver a la izquierda una imagen a escala de grises y a la derecha podemos ver una típica imagen a color, esta imagen fue tomada en los principios del desarrollo del proyecto, así viendo la capacidad de la cámara de profundidad. La imagen de la izquierda puede no ser muy clara para nosotros, pero esta imagen contiene la información de la distancia de los objetos respecto a la cámara, por ejemplo, la mano que está más cerca de la cámara aparece como negra, esto indica que está muy cerca de la cámara, por lo cual no puede ser leída, luego se puede apreciar que el cuerpo tiene un color medio blanco indicando una distancia, y la cabeza parece estar más lejana por su color gris, así como podemos ver que las persianas se muestran en un gris mucho más oscuro; una observación muy importante es que podemos visualizar que detrás del hombro izquierdo se tiene una línea completamente negra, esto se debe a que la luz se está reflejando a través de las persianas, así como la cuadricula de puntos infrarrojos, al ser reflejada y no retornada al sensor de profundidad este muestra esa parte como negra, y esto se puede interpretar de dos maneras, la primera, que el objeto se encuentra muy lejos del rango del proyector o la segunda, que existe algún tipo de reflejo, como por ejemple espejos o metal, y esto distorsiona la captura, mostrando esa parte como negro para la imagen de profundidad.

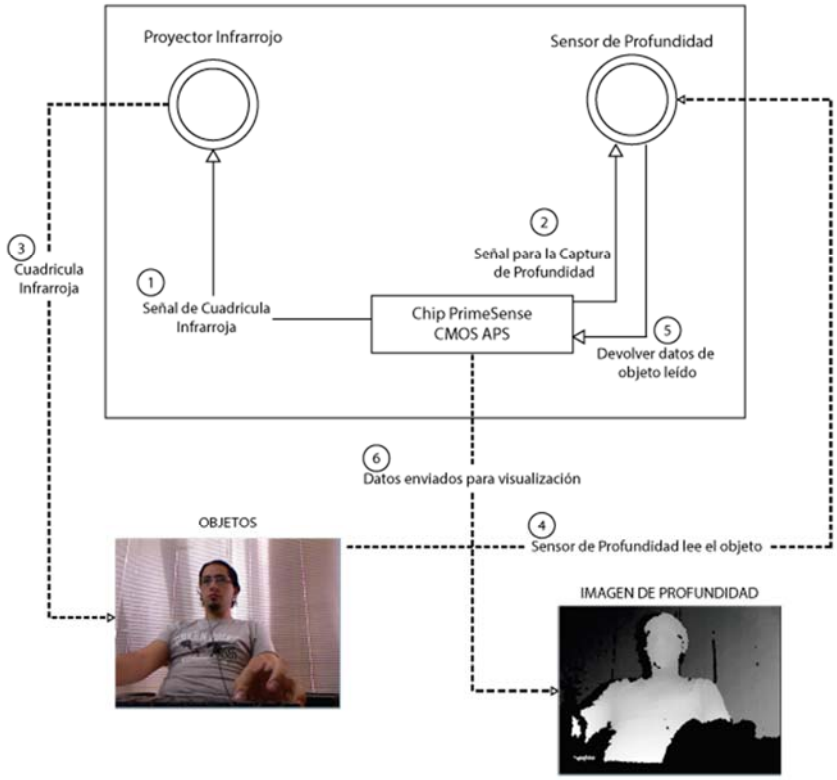

Figura 1-: Como se analiza los datos de profundidad
Fuente: Kinect for Windows SDK Programming Guide.9
Elaborado y Traducido por: David Zurita

En la figura 1-5 observamos el proceso de análisis de la cuadricula de puntos infrarrojos. Primero el chip CMOS APS, del que se habló en la sección anterior, manda la señal para el proyector de la cuadricula infrarroja sea proyectada, como segundo paso tenemos que se abre la señal para capturar la cuadricula y la profundidad, el tercer paso es la proyección de la cuadricula sobre los objetos al frente de la cámara, el cuarto paso es la obtención de la cuadricula con el sensor de profundidad, este devuelve los datos leídos de los objetos reflejados al chip CMOS APS como el quinto paso, y para finalizar el chip se encarga de procesar las imágenes obtenidas y estas imágenes son enviadas para la visualización. Este es el proceso que ocurre dentro del proyector infrarrojo y el sensor de profundidad para la obtención de imágenes de profundidad.

9 JANA, Abhijit. Kinect for Windows SDK Programming Guide, Birmingham Reino Unido: Packt Publishing, Diciembre 2012 1ra edición. 392 p.

Con esto podemos entender cómo trabajan conjuntamente el proyector infrarrojo y el sensor de profundidad para brindar imágenes de profundidad que son tratadas por los programadores.

1.2.3. Micrófonos:

Los micrófonos están localizados en la parte inferior del dispositivo, la función de estos es receptar el sonido que genera el usuario, con esto se obtiene comandos con voz para que la consola ejecute los comandos; existen varios micrófonos ubicados en el Kinect, creados para poder interpretar la distancia de los comandos, así como poder interpretar varios comandos de diferentes usuarios.

1.2.4. Motor:

En la base del Microsoft Kinect se encuentra un pequeño motor, su función es la rotación de la cabeza en donde están las cámaras y los micrófonos, con el fin de generar la posibilidad de poder colocar el dispositivo en cualquier lugar dentro de una sala, ya sea esto en una parte alta, como encima de una televisión, o en la base del mismo. Este motor tiene la capacidad de generar una rotación de 30 grados.

1.3. OpenNI.

La venta masiva de Kinects, presentó la iniciativa de ver más allá del funcionamiento del dispositivo dentro de los videos juegos, tanto así que la empresa Adafruit, empresa desarrolladora de software de Nueva York, ofreció un premio a quien sea capaz de crear un driver para conectar un Kinect a una computadora, el premio consistía de $2000 dólares americanos; Microsoft al leer de dicha recompensa público un aviso que el Kinect fue creado para el uso de Xbox, mas que no fue pensando para el uso con computadoras, y que tal premio no debería existir. Con este comentario de Microsoft, se logró subir el premio a $3000 dólares americanos, motivando así a más entusiastas a crear un driver para el uso con una computadora. Se creó una comunidad llamada OpenKinect que empezó a trabajar en este driver, pero Hector Martin con la ayuda de PrimeSense, empresa creadora del chip que maneja la cámara de profundidad del Kinect, reclamaron el premio, logrando conectar el Kinect con una computadora con sistema operativo Linux y se logró conectar la cámara

Kinect y utilizar su cámara de profundidad.[10] Esto demuestra lo que las comunidades pueden lograr.

Esta alianza, quienes reclamaron el premio, decidieron continuar con su trabajo, para esto crearon OpenNI, en donde NI es acrónimo de Natural Interaction, esto refiere a las nuevas opciones de interfaces para los usuarios, en donde se desea eliminar los periféricos comunes de hoy y que nuestra voz y gestos sean los que manejen el ordenador.

OpenNI fue creada en noviembre del 2010, con la visión de generar una librería que sea capaz de utilizar las habilidades del Microsoft Kinect y poner estas a manos de los desarrolladores que deseen crear aplicaciones las cuales cambien la manera de pensar cómo se hacen las cosas hoy en día, que estas faciliten la interacción entre el usuario y la computadora. Ya que OpenNI tiene aportes de PrimeSense, es una biblioteca y drivers con visión a futuro, ya que la compañía está creando nuevos dispositivos con cámaras de profundidad, lo cual al usar esta librería nos da la posibilidad de mantener el código ya creado para el Kinect para los nuevos dispositivos que vayan saliendo al mercado.

OpenNI provee dos paquetes de librerías, la primera maneja la captura de la profundidad y el análisis de esta, posee la licencia LGPL, la cual es completamente libre; el segundo paquete maneja el análisis de esqueleto para los usuarios y gestos, este posee un tipo de licencia comercial, esta no permite la distribución de las aplicaciones creadas, excepto que las aplicaciones tengan motivos educativos; al segundo paquete se lo conoce como NITE, parte de la librería que maneja todas las articulaciones del usuario y sus respectivas interpretaciones en código.

Hasta la fecha 1 de diciembre del 2013 OpenNI maneja la versión 1.96 para Processing, esta versión fue actualizada en agosto del 2013, brindando la facilidad al instalar la librería en la computadora, sea cual sea el sistema operativo que se tenga, y brindando nuevas actualizaciones a las librerías.

[10] BORENSTEIN, Greg. Making Things See: 3D vision with Kinect, Processing, Arduino, and MakerBot. Estados Unidos de Norte America: O'Reilly**,** Febrero 2012, 1ra edición. 440 p

1.4. Processing.

Processing es un lenguaje creado por Casey Reas y Benjamin Fry, en el grupo de Estética y Computación en el laboratorio de Medios en MIT, la idea surgió en la primavera del 2001 cuando Casey, un artista conceptual y programador, hablaba con Benjamin, un experto en la visualización de datos; hablaban lo complicado que se ha vuelto la creación de aplicaciones visuales con los lenguajes utilizados durante esos años, de allí surgió la idea de crear un nuevo lenguaje que sea fácil de aprender y que de la misma manera sea sencillo la programación visual y gráfica[11]; se obtuvo una versión alfa del lenguaje en agosto del 2002, esta se mantuvo en esta versión hasta abril del 2005, la cual se manejaba dentro de las clases del MIT, luego se la publicó como versión beta hacia el público, esto hasta el 2008. Durante este tiempo la influencia de entusiastas al ver la potencia del lenguaje fue creciendo, permitiendo la creación de librerías que extienden la funcionalidad de Processing, hoy en día hay más de 100 librerías, dentro de estas existen las funcionalidades de 3d, animación, geometría, hardware, matemáticas, simulación, sonido, tipografía, video, red, etc. Hasta la fecha 1 de diciembre del 2013 Processing se encuentra en la versión 2.1, siendo esta la más estable, esta versión es capaz de extender su funcionalidad no solo a plataformas de escritorio, sino también a la web con Processing.js que es una extensión para los navegadores, así como para Android y para el control de hardware Arduino.

Processing es basado en Java como su lenguaje, lo cual permite que este sea ejecutado en cualquier sistema operativo sin ninguna restricción, de hecho Processing es una subclase de PApplet de Java, este permite la traficación de objetos con mucha facilidad. El objetivo principal de este lenguaje es mostrar a gente que no conoce sobre programación lo sencillo que puede ser crear sus propias aplicaciones, así como para la gente que ya conoce, brinda la capacidad de realizar programas robustos minimizando el tiempo de codificación y manejando la visualización de datos y la visualización grafica de una manera muy sencilla y efectiva.

[11] REAS, Casey; FRY, Ben. Getting Started with Processing. Estados Unidos de Norte America: O'Reilly Media, Julio 2010, 1ra edición. 210 p.

Figura 1-: IDE Processing
Fuente: Captura de pantalla
Elaborado por: David Zurita

Para mantener esta simplicidad en el código, se creó una interfaz o IDE para la codificación de Processing, este consta solo de una pantalla o como se lo llama en Processing un Sketch, en donde se inserta el código, siendo súper sencillo y claro, ya que no tiene nada de herramientas extras dentro de la ventana, es solo el ambiente de desarrollo y la persona que desea codificar. Una de las ventajas de ser basado en Java es la capacidad de poder crear objetos, así manteniendo la seguridad y robustez del lenguaje.

Siendo de licencia GPL y LGPL, brinda las mejores opciones para la codificación libre, así como ha generado una comunidad que ha tomado la iniciativa de crear distintas aplicaciones interactivas, como mencionamos en el punto anterior, gracias a estas comunidades se creó la extensión para Microsoft Kinect, OpenNI, esta extensión o librería es la que se explota para la creación de esta disertación de grado.

1.5. Motivación de la creación del juego.

En el antiguo Japón en una era de constantes guerras, se desarrolló varios artes marciales, en donde se enseñaba el cómo defenderse de los atacantes, el uso de armas, entre otras cosas; estas artes marciales han pasado de generación en generación hasta nuestros días; sus enseñanzas milenarias se las dictaba en un sitio conocido como Dojo, un sito únicamente dedicado al maestro donde enseñaba al alumno las diferentes técnicas, y así el paso de sabiduría se establecía, este Dojo poseía muchas de las veces, un altar del Shintoismo en donde se quemaban velas para purificar el ambiente, el Dojo era un cuarto adecuado para enseñar técnicas, aquí también se poseía las armas con las cuales se aprendían dichas técnicas. Estas enseñanzas han pasado por siglos de maestros y alumnos, mejorando las técnicas y dando un toque personal a estas.

Hoy en el mundo que vivimos es un mundo globalizado con la tecnología al alcance de casi todos, viviendo en una era del conocimiento, estas técnicas y artes marciales antiguas se deben adaptar a este nuevo mundo, es por ello que se ha pensado y desarrollado el unificar estos conceptos antiguos con la tecnologías, así enseñando los cortes básicos de espada japonesa a un juego. Con el fin de fusionar la idea milenaria de un Dojo con la tecnología actual, hemos simulado un Dojo virtualmente, brindando así la idea de un sitio específico para enseñar las técnicas en donde el jugador decida jugar. Fácilmente se podrá identificar claramente los elementos de un dojo, ya mencionados anteriormente, también tendremos el maestro, quien nos enseñará. Claro que al ser un juego este toma una modalidad más informal y presentamos a un viejo maestro, tan viejo que hasta cuernos le han salido de la cabeza, así dando un tono de juego en donde el usuario estará aprendiendo los cortes básicos de espada, pero a su vez entenderá como era antiguamente las enseñanzas.

Capítulo II Herramientas y Metodología

En el presente capítulo se describirá las herramientas que hicieron posible el desarrollo de la presente disertación de grado.

Se ha tomado en cuenta tanto las herramientas de software como la metodología para el desarrollo.

2.1. Extreme Programming.

Es una metodología de desarrollo de software muy conocida, en donde la idea principal es obtener los requerimientos sobre lo que se va a desarrollar, crear ciclos pequeños de desarrollo, una vez concluido el ciclo, se realizan pruebas y se genera una parte del programa total que ya puede ser puesto a distribución. Este modelo es excelente para la presente disertación ya que da la flexibilidad de dividir en diferentes partes el producto total, que en este caso es el juego, al poder dividir estas partes se pueden ir creando estas individualmente, lo cual lleva a un producto con la certeza de su funcionalidad.

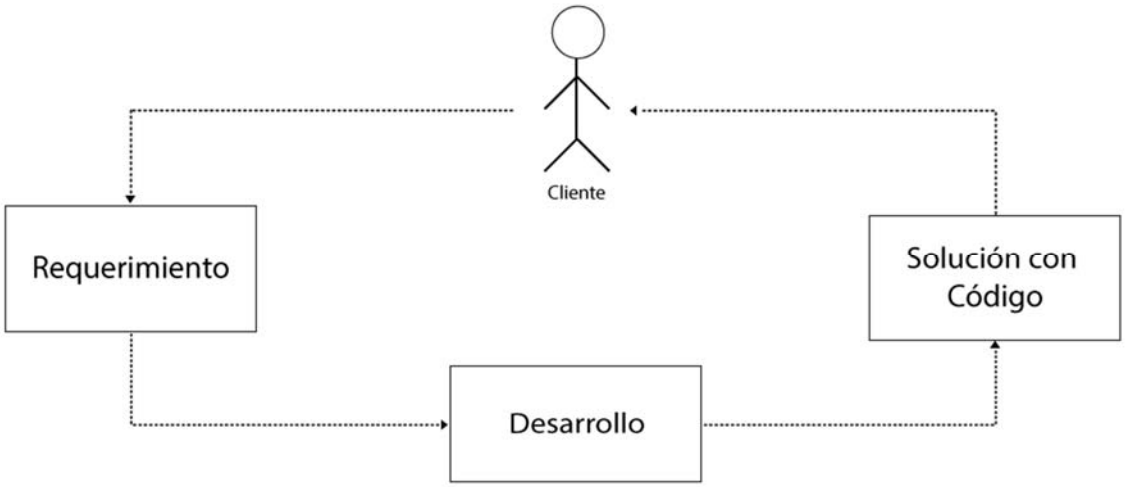

Figura 2-1: Ciclo de Extreme Programming
Fuente: Extreme Programming Pocket Guide[12]
Elaborado Traducido por: David Zurita.

Como podemos apreciar en la figura 2-1, tenemos un ciclo de extreme programming, lo que se analiza es un solo requerimiento, este se lo específica, se lo desarrolla y se produce una solución con código, una vez hecho esto está listo para ser entregado al cliente; dado el caso de esta disertación no existe un cliente como tal, pero se realizan las pruebas respectivas cada requerimiento desarrollado, de esta manera se asegura que el producto que se está creando funcione adecuadamente y que todas sus funcionalidades sean creadas.

Por lo cual se debe enlistar los requerimientos que nuestro desarrollo llevará; para el desarrollo de un juego interactivo con una cámara Microsoft Kinect para el reconocimiento del jugador y los diferentes cortes de espada.

[12] CHROMATIC. *Extreme Programming Pocket Guide.* Estados Unidos de Norte América: O'Reilly Media, Junio 2009, Ebook. 108 p.

2.1.1. Requerimientos:

2.1.1.1. Nube de Puntos:

Es el espacio en tres dimensiones a utilizar para el análisis por donde se van a realizar los cortes. Este espacio es el examinado por la cuadricula de puntos infrarrojos que se proyecta del Microsoft Kinect.

2.1.1.2. HotPoint.

Espacio definido en las tres dimensiones el cual será utilizado para el análisis de los cortes. Estos son definidos con figuras geométricas ya que es la forma más sencilla de ubicarlos en el espacio, y de graficarlos para su visualización.

2.1.1.3. Activación del HotPoint.

Los HotPoints al ser declarados ocupan un espacio en la nube de puntos, este espacio al ser definido puede ser analizado, de existir algún cambio dentro de este espacio se lo considerará el paso de un corte, con esto se empezará a analizar los diferentes cortes y espacios necesarios.

2.1.1.4. Cortes con espada.

Basándonos en antiguas escuelas de artes marciales de Japón, podemos sacar los seis diferentes tipos de cortes que se realizaban, estos nos servirán como base para realizar los análisis de los cortes. Los seis cortes son:

- 2 diagonales llamados Keza Kiri derecho e izquierdo de arriba hacia abajo.
- 2 diagonales llamados Age Kiri derecho e izquierdo de abajo hacia arriba.
- 1 horizontal llamado Do kiri.
- 1 vertical llamado Men kiri.

2.1.1.5. Ubicaciones de HotPoints para cortes.

Una vez desarrollado un HotPoint, este se debe replicar para poder tener los espacios definidos para los cortes que se van a realizar, para esto se deberá crear una clase con HotPoints para la optimización de la creación de estos objetos, una vez creados se los debe ubicar dentro del espacio para que este pueda ser leído y analizado, y con esto se pueda dar a conocer que tipo de corte se está realizando.

2.1.1.6. Integración gráfica.

Se debe conocer cómo se realizará la integración de los objetos gráficos entregados por el ilustrador, estos objetos deben formar parte del concepto para el juego, y deben ser integrados a los HotPoints ya creados.

2.1.1.7. Unificación con animaciones.

Una vez colocada la gráfica, se debe unificar las animaciones con los analices de los HotPoints, estas animaciones deben ser respecto al corte y deben ser integradas con la animación del maestro que informa al usuario el nombre del corte que está realizando.

Al concluir con estos requerimientos se tiene el juego completo. El desarrollo y las pruebas que se irán realizando se colocaran en el capítulo 4; y la arquitectura del proyecto se lo colocará en el capítulo 3 debido a su extensión.

Requerimientos	Descripción
Nube de Puntos	Área virtual donde se va a trabajar
HotPoint	Espacio vectorial para el análisis
Activación del HotPoint	Análisis de activaciones de nubes de puntos
Cortes con espada	Seis cortes básicos
Ubicaciones de HotPoints para cortes	Espacio de ubicación para HotPoints
Integración gráfica	Colocar la gráfica sobre los HotPoints
Unificación con animaciones	Validación de cortes y representación por animación

Tabla 2-: Requerimientos y sus respectivas descripciones.
Elaborado por: David Zurita.

2.2. Processing.

Como se explicó en el marco teórico, Processing es un lenguaje de programación, está basado en Java, pero al final del día tiene sus propias palabras reservadas y sus propias variables; en este capítulo se explicará cómo funciona Processing, y como es capaz de integrar librerías extras para extender su funcionalidad.

2.2.1. IDE

A continuación se explicará cómo está compuesto el IDE para programar en Processing.

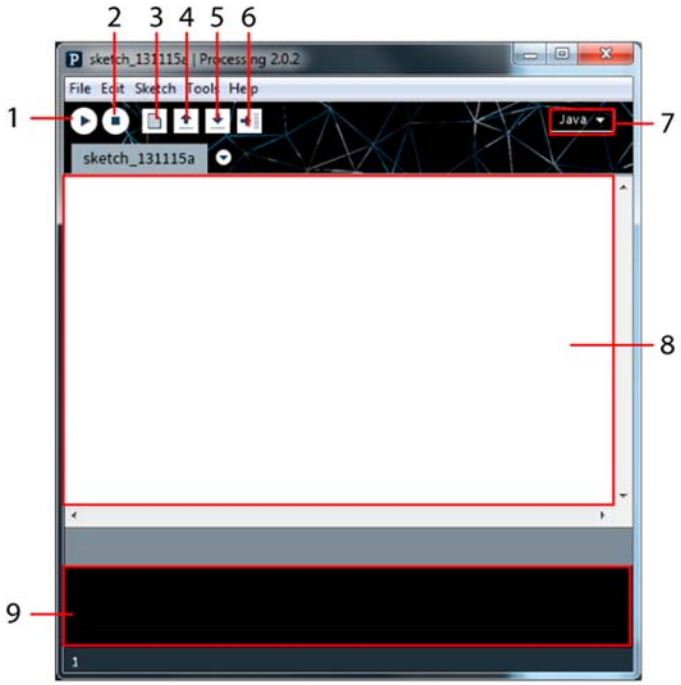

Figura 2-2: IDE Processing
Fuente: Captura de pantalla
Elaborado por: David Zurita

1. Botón de correr o “Run”, este botón compila y ejecuta el código que se está realizando, de existir un error, lo mostrará en la consola.

2. Botón de detener o “Stop”, como lo indica este detiene la aplicación que se encuentra corriendo. Es muy útil al realizar pruebas y debido a algún error la aplicación se cuelga, este cierra la ventana de la aplicación.

3. Nuevo o “New”, este nos ayuda a crear un Nuevo sketch.

4. Abrir o “Open”, este abre una ventana, en la carpeta donde se encuentra el último proyecto abierto.

5. Grabar o “Save”, aquí es donde se indica al IDE para que grabe la aplicación que se está realizando, al ser la primera vez se debe indicar la carpeta en donde se desea grabar la aplicación.

6. Exportar o “Export”, este es el botón más útil del IDE, ya que con este se puede exportar la aplicación con un solo clic, las opciones de exportación son para Windows, Mac OS X, o Linux, la exportación se puede hacer para los tres sistemas operativos o para uno solo, dependiendo de las necesidades. Al realizar la exportación en la carpeta del proyecto se crearán los respectivos ejecutables.

7. Este indica en que se está programando como base, Java es el estándar, pero si se requiere programar para Android, aquí se lo puede especificar para que el código y la compilación sea realizada para dicho dispositivo, así como también hay como extender la programación hacia Processing.js, que es una librería de JavaScript, en donde el código estaría creado en JavaScript y listo para ser ejecutado en la web; también se brinda la opción de programar para iOS, en donde se hace un puerto entre Java, este último todavía se encuentra en etapa beta, así como el código para Android.

8. Sketch, aquí es donde irá el código que va a tener la aplicación.

9. Consola, como casi todo IDE este cuenta con la consola, en donde se pueden imprimir resultados de cálculos u observar los errores que se vayan generando al codificar para la aplicación.

Botón	Nombre	Descripción
1	Correr	Corre el Sketch
2	Detener	Detiene el Sketch
3	Nuevo	Crean un nuevo Sketch
4	Abrir	Abre un Sketch
5	Grabar	Graba el Sketch
6	Exportar	Exporta el Sketch
7	Código	Código en el que se compila el Sketch
8	Sketch	Espacio para codificar
9	Consola	Administrador de texto

Tabla 2-: Componentes Processing
Elaborado por: David Zurita.

2.2.2. Funciones:

A continuación se explicará las funciones básicas de Processing, estas se encuentran presentes en la mayoría de sketch debido a su funcionalidad.

2.2.2.1. setup ():

El código dentro de la función setup se ejecuta una sola vez, cuando el programa se corre por primera vez; se utiliza para la inicialización de variables[13].

2.2.2.2. draw ():

Una vez ejecutada la función setup, el código dentro de draw es el código que se corre continuamente, este código es el que se utiliza para mostrar animaciones o código interactivo. Es una función súper importante ya que se ejecutara mientras el programa corra, esto quiere decir que todos los cálculos ejecutados aquí irán variando conforme el programa siga corriendo.[14]

2.2.2.3. frameRate ():

Esta función se la debe localizar dentro del setup, y lo que garantiza es la cantidad de imágenes que se van a ver por segundo mientras la aplicación se esté corriendo.

[13] REAS, Casey; FRY, Ben; MAEDA, John. Processing: A Programming Handbook for Visual Designers and Artists. Estados Unidos de Norte America: MIT Press, Agosto 2007, 1ra edición. 736p.
[14] Id

2.2.2.4. size (x, y, z):

Especifica el tamaño de la ventana de aplicación, sus parámetros son la distancia en pixeles en x así como en y, y de ser en 3 dimensiones, se especifica la librería gráfica a utilizar.

Funciones	Descripción
setup()	Configuración de Variables para el uso del programa
draw()	Función que se refresca por el Frame Rate, y despliega en la pantalla el sketch
frameRate()	Especifica la velocidad de cuadros por segundo a mostrar
size()	Declara el tamaño de la pantalla del sketch

Tabla 2-: Funciones de Processing
Elaborado por: David Zurita.

2.2.3. Variables:

Ya que Processing es basado en Java, mantiene las variables principales, pero también posee variables específicas para el manejo de entidades 3d, imágenes y videos. A continuación haremos una explicación de cómo funcionan estas variables.

2.2.3.1. Movie:

Es un tipo de dato creado por Processing para almacenar y reproducir películas de formato QuickTime; el cual brinda la posibilidad de reproducir la película, detenerla, mantenerla en un bucle. Antes de poder reproducir la película se de leer dentro de la variable tipo movie, una vez hecho esto se puede manejar la variable.

2.2.3.2. PImage:

Tipo de dato para almacenar imágenes, la P representa Processing, este tipo de dato es capaz de almacenar imágenes: gif, jpg, tga y png; la variable debe ser inicializada con loadImage ()[15].

2.2.3.3. PVector:

Clase creada por Processing para describir un vector geométrico de dos o tres dimensiones; este almacena los componente del vector en x, y y z.[16]

[15] Id
[16] Id

Variables	Descripción
Movie	Guarda en la variable un archivo tipo película
PImage	Guarda una imagen
PVector	Almacena un vector en tres dimensiones

Tabla 2-: Variables de Processing

Elaborado por: David Zurita.

2.3. OpenNI.

OpenNI es el puerto de comunicación entre el Microsoft Kinect y el código que se va a generar, para el proyecto que se está desarrollando se posee una librería específica para Processing, esta debe ser siempre importada dentro de cada sketch para manejar la cámara de profundidad.

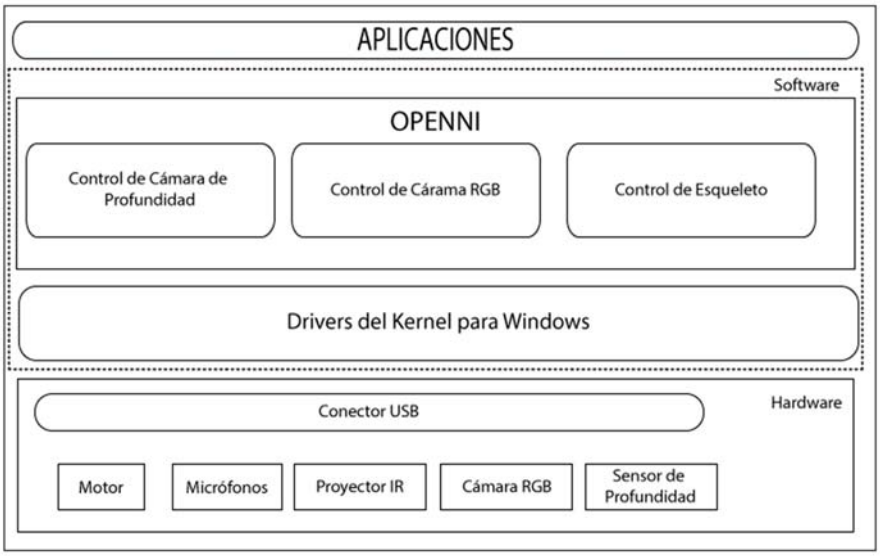

Figura 2-: Arquitectura SDK OpenNI.
Fuente: Kinect in Motion – Audio and Visual Tracking by Example [17].
Elaborado y Traducido por: David Zurita.

La arquitectura que presenta la librería de OpenNI es la que se puede observar en la figura 10, esta es el paso entre el hardware y el código a ser desarrollado dentro de Processing. Como se puede observar el SDK, maneja el control de la cámara de profundidad, es decir tanto el proyector IR, así como del sensor de profundidad; esto también permite que maneje el esqueleto del usuario; y la última funcionalidad es el manejo de la cámara RGB. El SDK hasta la fecha del desarrollo de la presente disertación no posee el control ni los micrófonos instalados alrededor del Kinect o del motor ubicado en la base de este.

Esta arquitectura trae consigo una clase que maneja todos las características previamente dichas, a continuación de describirá esta clase, con sus respectivas funciones, las cuales hacen posible el desarrollo de la presente disertación.

[17] GIORIO, Clemente; FASCINARI, Massimo. Kinect in Motion – Audio and Visual Tracking by Example, Reino Unido: Packt Publishing, Abril 2013 1ra edición. 112 p.

2.3.1. SimpleOpenNI:

La clase SimpleOpenNI crea un objeto tipo SimpleOpenNI, este objeto es el que se utiliza para controlar la cámara de profundidad, al inicializarlo este requiere el paso del sketch con el que se va a trabajar.

Para manejar la cámara RGB utiliza la función enableRGB (), y para la cámara de profundidad se utiliza enableDepth.

Algo que siempre se debe colocar es la función update () ya que esta refresca la cámara y se pasen nuevos datos, esto depende de la resolución con la que se está trabajando.

Para trabajar con la nube de profundidad se debe utilizar depthMapRealWorld (), este devuelve un arreglo de puntos en tres dimensiones, para esto se crea un arreglo de PVector, para poder manejar los datos que devuelve esta función, esta es la que se encarga de ver la profundidad de los objetos y ver si dentro de esto existe algún cambio. Este arreglo de puntos tridimensionales es donde se captura en cada foto por segundo los cambios que se realizan en la cuadricula de puntos infrarrojos que se genera, de esta manera se puede buscar los cambios dentro de esta cuadricula; esta es la función más importante con la que se va a trabajar ya que devolverá todos los datos para ser analizados.

Capítulo III Diseño de Aplicación

En el presente capítulo se especificará las funcionalidades de la aplicación y como se procederá con estas; todas estas funcionalidades son basadas en los requerimientos obtenidos al usar Extreme Programming como la metodología del proyecto. Estas funcionalidades dan paso a los diagramas de UML, en donde se especifica la arquitectura que poseerá el juego.

3.1. Planificación

Aquí se describe las funcionalidades y sus flujos que podrá tener el usuario mientras este dentro del juego.

3.1.1. Funcionalidades:

Basándonos en los requerimientos obtenidos debemos tener siete funcionalidades dentro del juego, y dentro de estas existen seis extras que son las visualizadas por el usuario, estas son los cortes que podrá realizar dentro de la aplicación. Por lo tanto estas seis últimas son las que se generarán como funcionalidades para el usuario, presentando los diferentes casos y flujos. Estas son:

- Corte izquierdo de arriba hacia abajo, llamado Hidari Keza Kiri.
- Corte derecho de arriba hacia abajo, llamado Migi Keza Kiri.
- Corte izquierdo de abajo hacia arriba, llamado Hidari Age Kiri.
- Corte derecho de abajo hacia arriba, llamado Migi Age Kiri.
- Corte horizontal, llamado Do Kiri.
- Corte vertical, llamado Men Kiri.

Dirección Corte	Nombre en Japonés
Izquierda de arriba hacia abajo	Hidari Keza Kiri
Derecha de arriba hacia abajo	Migi Keza Kiri
Izquierdo de abajo hacia arriba	Hidari Age Kiri
Derecha de abajo hacia arriba	Migi Age Kiri
Horizontal	Do Kiri
Vertical	Men Kiri

Tabla 3-: Cortes.
Elaborado por: David Zurita.

Con el fin de comprender como se va a realizar la evaluación de los espacios para los cortes se ha realizado un gráfico explicando los espacios en donde se van a encontrar los HotPoints dentro del juego.

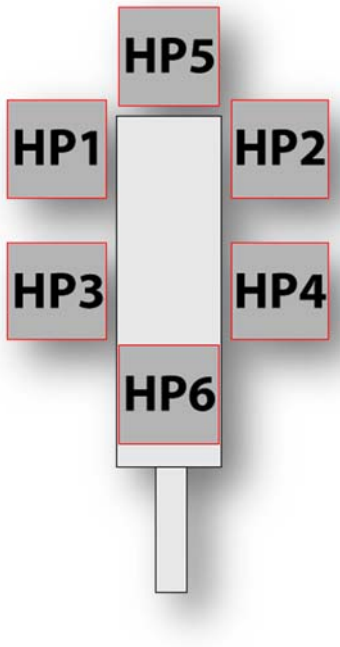

Figura 3-: Gráfico de la ubicación de los HotPoints.
Elaborado por: David Zurita.

La figura 11 nos indica la ubicación de los HotSpots del juego, estos son los espacios que serán evaluados dentro del juego para conocer el corte que se está realizando por el usuario, esto es importante ya que para entender los flujos principales se explicará el

procedimiento a ser realizado cuando se realiza el corte en específico. La nomenclatura de HP se refiere a los HotPoints ubicados alrededor del objetivo a ser cortado, estos son cinco, para analizar los seis cortes.

3.1.2. Flujos Principales:

A continuación se presentan los flujos principales que tendrá el jugador al utilizar la aplicación.

Estos se analizan con los requerimientos ya especificados, en donde se crean un cuadro para describir las acciones que toma el actor y el programa, estas acciones reflejan todos las posibles respuestas que tiene el programa a los diferentes movimientos que va a tener el usuario, también describiendo los posibles errores, y los mensajes de error que estos tendrán. Esto ayuda para la creación de diagramas más adelante, para tener claro como estos fueron realizados.

3.1.2.1. F0: Iniciar Aplicación:

3.1.2.1.1. Flujo Principal:

Paso	Actor	Paso	Sistema	Excepción
1	Abre la Aplicación	2	Inicia la aplicación	E1
		3	Carga parte gráfica	
		4	Aplicación lista para el Actor	

3.1.2.1.2. Excepciones:

Código	Descripción	Acción
E1	Cámara Kinect no conectada	Mostrar Error y solicitar conectar cámara

3.1.2.2. F1: Corte izquierdo de arriba hacia abajo:

3.1.2.2.1. Flujo Principal:

Paso	Actor	Paso	Sistema	Excepción
1	Se coloca en frente de la cámara Kinect			
2	Realiza el movimiento del corte izquierdo de arriba hacia abajo	3	Reconocimiento de paso de objeto por HotPoint 1	
		4	Reconocimiento de paso de objeto por HotPoint 4	E2
		5	Animación de corte Hidari Keza Kiri	
		6	Animación de Maestro Hidari Keza Kiri	

3.1.2.2.2. Excepciones:

Código	Descripción	Acción
E2	No se realiza el corte	El juego reinicia variables

3.1.2.3. F2: Corte derecho de arriba hacia abajo:

3.1.2.3.1. Flujo Principal:

Paso	Actor	Paso	Sistema	Excepción
1	Se coloca en frente de la cámara Kinect			
2	Realiza el movimiento del corte derecho de arriba hacia abajo	3	Reconocimiento de paso de objeto por HotPoint 2	
		4	Reconocimiento de paso de objeto por HotPoint 3	E2
		5	Animación de corte Migi Keza Kiri	
		6	Animación de Maestro Migi Keza Kiri	

3.1.2.3.2. Excepciones:

Código	Descripción	Acción
E2	No se realiza el corte	El juego reinicia variables

3.1.2.4. F3: Corte izquierdo de abajo hacia arriba:

3.1.2.4.1. Flujo Principal:

Paso	Actor	Paso	Sistema	Excepción
1	Se coloca en frente de la cámara Kinect			
2	Realiza el movimiento del corte izquierdo de abajo hacia arriba	3	Reconocimiento de paso de objeto por HotPoint 3	
		4	Reconocimiento de paso de objeto por HotPoint 2	E2
		5	Animación de corte Hidari Age Kiri	
		6	Animación de Maestro Hidari Age Kiri	

3.1.2.4.2. Excepciones:

Código	Descripción	Acción
E2	No se realiza el corte	El juego reinicia variables

3.1.2.5. F4: Corte derecho de abajo hacia arriba:

3.1.2.5.1. Flujo Principal:

Paso	Actor	Paso	Sistema	Excepción
1	Se coloca en frente de la cámara Kinect			
2	Realiza el movimiento del corte derecho de abajo hacia arriba	3	Reconocimiento de paso de objeto por HotPoint 4	
		4	Reconocimiento de paso de objeto por HotPoint 1	E2
		5	Animación de corte Migi Age Kiri	
		6	Animación de Maestro Migi Age Kiri	

3.1.2.5.2. Excepciones:

Código	Descripción	Acción
E2	No se realiza el corte	El juego reinicia variables

3.1.2.6. F5: Corte horizontal:

3.1.2.6.1. Flujo Principal:

Paso	Actor	Paso	Sistema	Excepción
1	Se coloca en frente de la cámara Kinect			
2	Realiza el movimiento del corte derecho de abajo hacia arriba	3	Reconocimiento de paso de objeto por HotPoint 4	
		4	Reconocimiento de paso de objeto por HotPoint 3	E2
		5	Animación de corte Do Kiri	
		6	Animación de Maestro Do Kiri	

3.1.2.6.2. Excepciones:

Código	Descripción	Acción
E2	No se realiza el corte	El juego reinicia variables

3.1.2.7. F6: Corte vertical:

3.1.2.7.1. Flujo Principal:

Paso	Actor	Paso	Sistema	Excepción
1	Se coloca en frente de la cámara Kinect			
2	Realiza el movimiento del corte vertical	3	Reconocimiento de paso de objeto por HotPoint 5	
		4	Reconocimiento de paso de objeto por HotPoint 6	E2
		5	Animación de corte Men Kiri	
		6	Animación de Maestro Men Kiri	

3.1.2.7.2. Excepciones:

Código	Descripción	Acción
E2	No se realiza el corte	El juego reinicia variables

3.2. Diseño

Para el diseño del juego se ha utilizado diagramas UML, Unified Modeling Languaje, por sus siglas en inglés, este lenguaje de modelado permite visualizar y especificar las funciones antes mencionadas y crear “planos” para la programación.

Se ha dividido en los diferentes diagramas para la mejor visualización de estos.

3.2.1. Diagramas de estructura.

Estos diagramas destacan las características que van a estar presentes en el juego.

3.2.1.1. Diagrama de clases

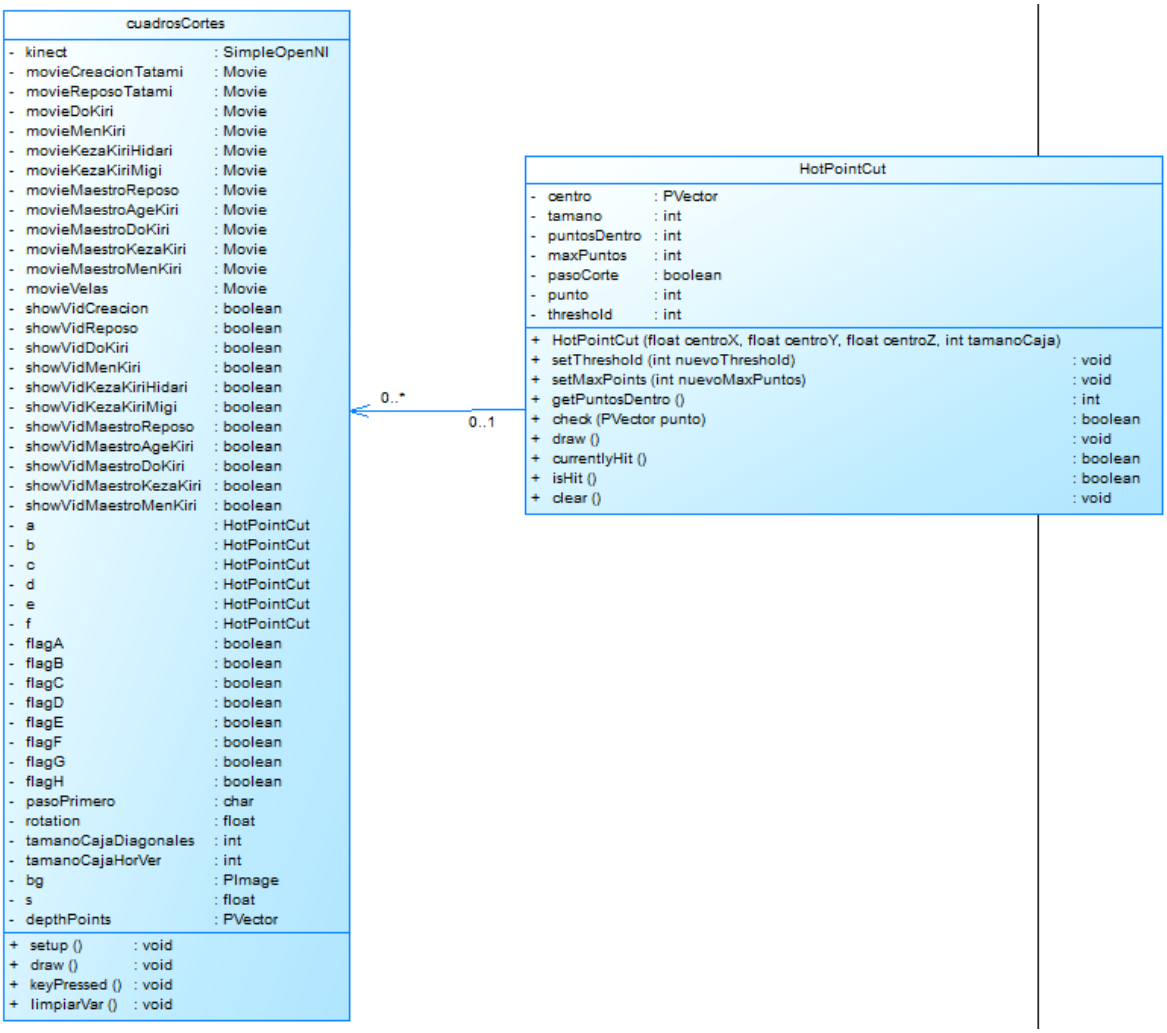

Figura 3-: Diagrama de Clases.
Elaborado por: David Zurita.

3.2.2. Diagramas de comportamiento.

3.2.2.1. Diagrama de procesos

El presente diagrama especificará las actividades a ser realizadas por el sistema, es decir, el enfoque del diagrama es cómo funcionará el sistema ante la presencia del usuario.

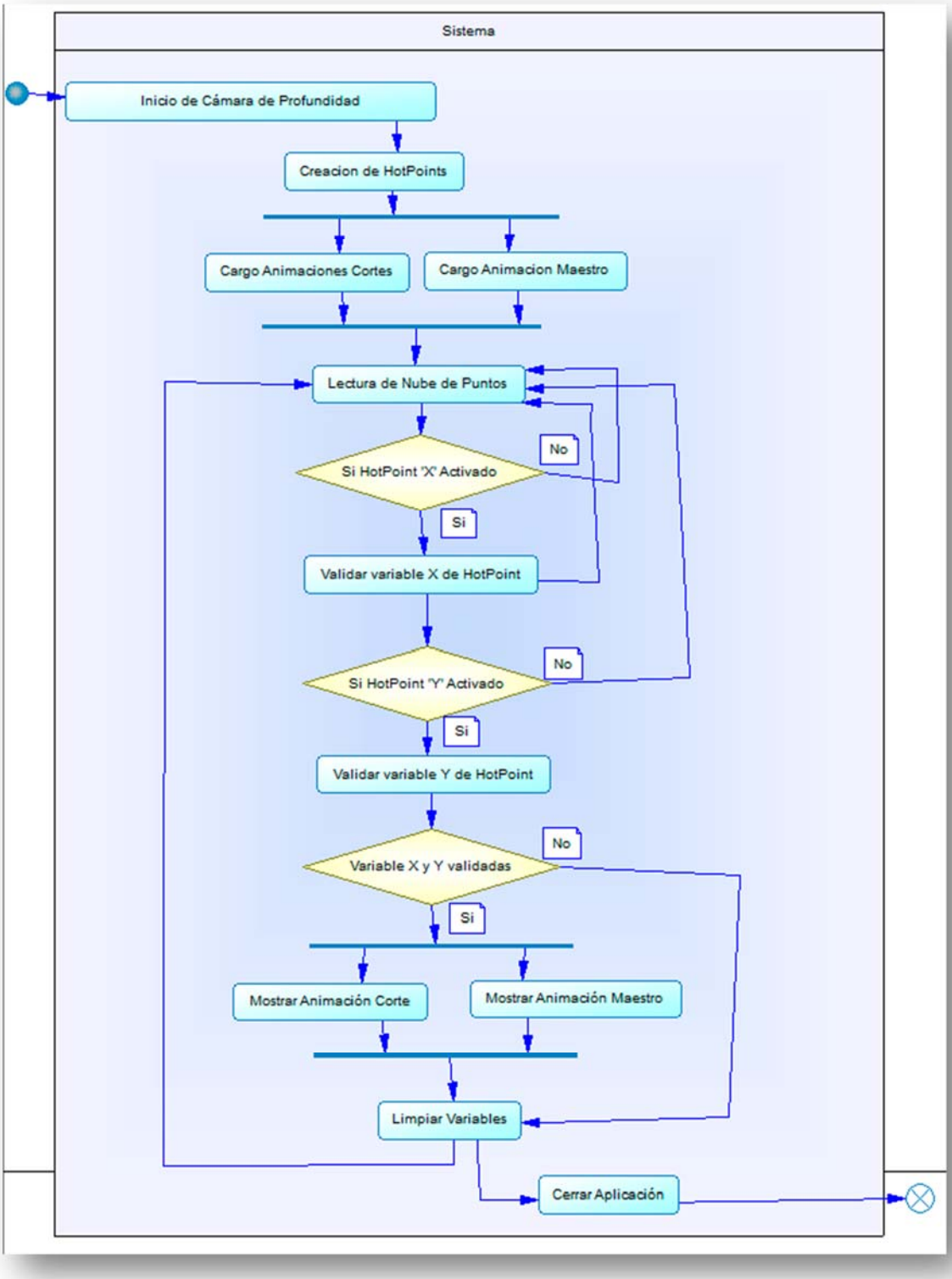

Figura 3-: Diagrama actividades.
Elaborado por: David Zurita.

3.2.3. Diagramas de interacción.

3.2.3.1. Diagrama de secuencia

Los presentes diagramas muestran el flujo de las funcionalidades ya explicadas anteriormente, estos son la base de la programación en cuanto a la respuesta que tendrá la aplicación ante los movimientos del usuario; se especificará cada funcionalidad con un diagrama.

3.2.3.1.1. F0: Iniciar Aplicación:

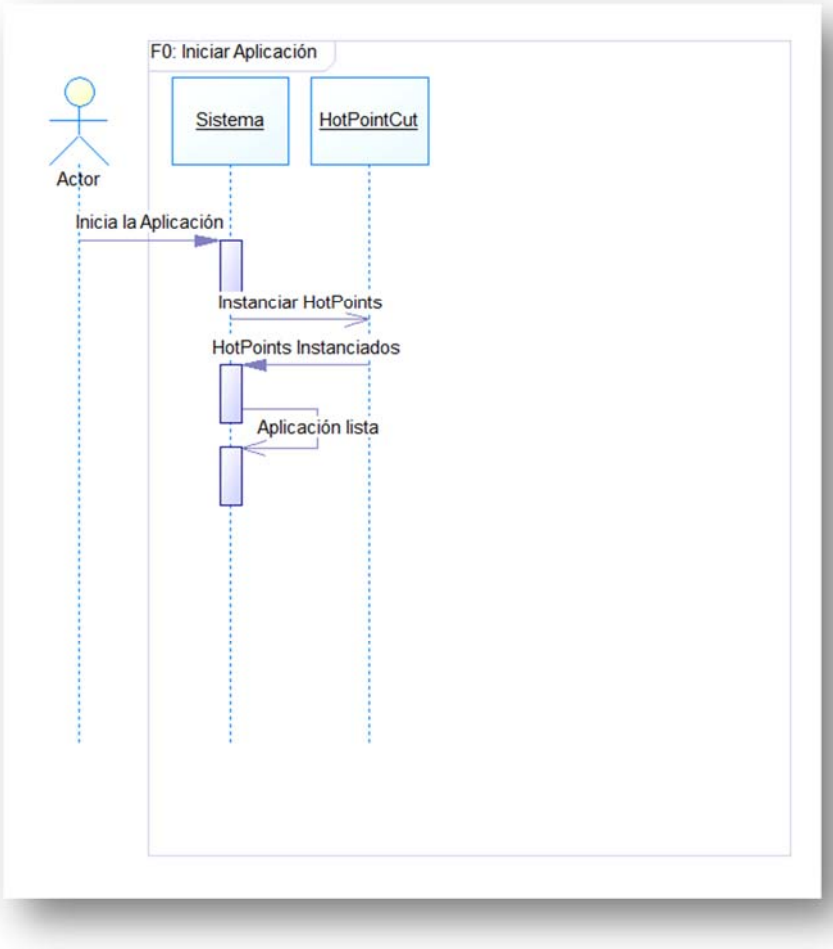

Figura 3-: Diagrama de Secuencia de Inicio de la aplicación.
Elaborado por: David Zurita.

3.2.3.1.2. F1: Corte izquierdo de arriba hacia abajo:

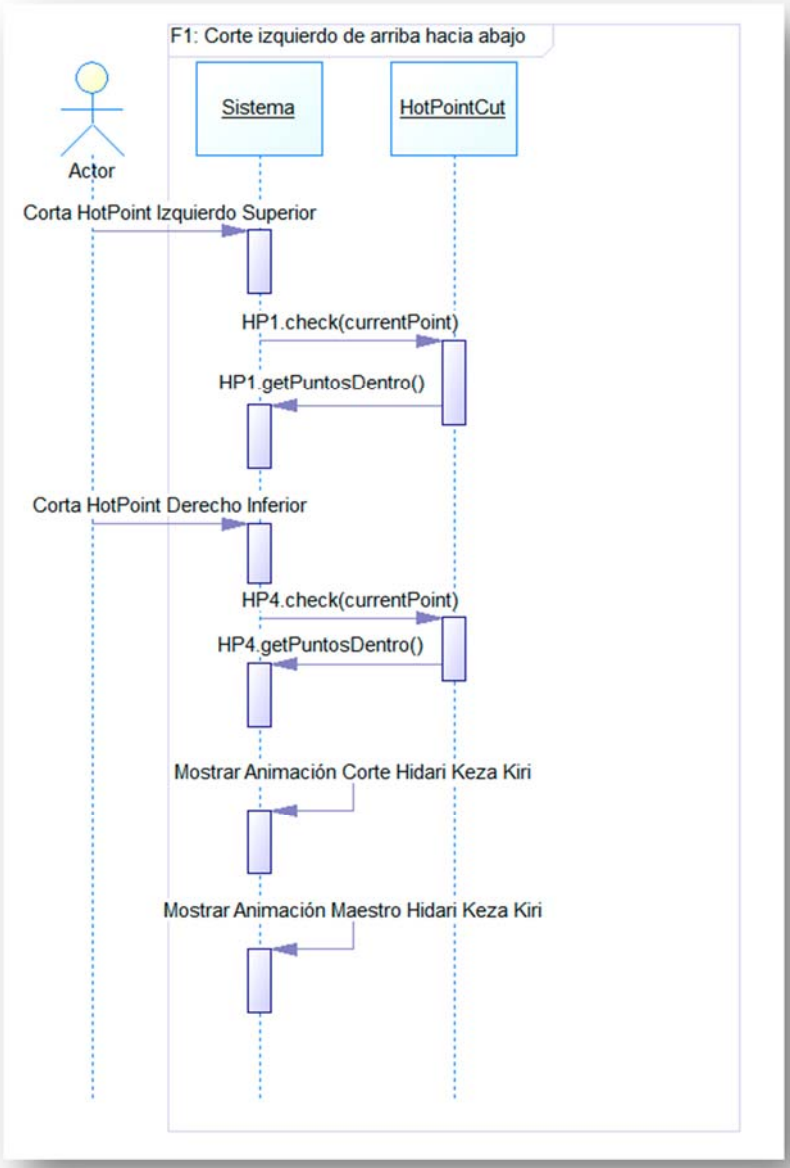

Figura 3-: Diagrama de Secuencia corte izquierdo de arriba hacia abajo.
Elaborado por: David Zurita.

3.2.3.1.3. F2: Corte derecho de arriba hacia abajo:

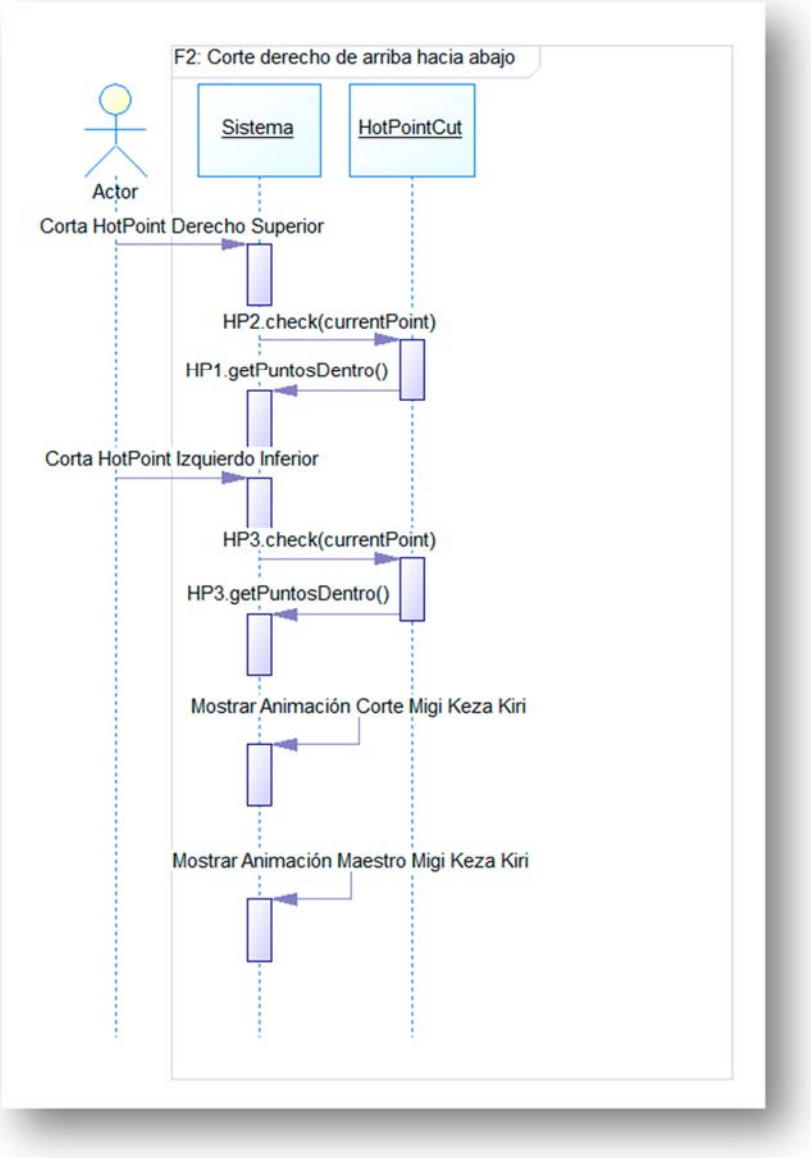

Figura 3-: Diagrama de Secuencia corte derecho de arriba hacia abajo.
Elaborado por: David Zurita.

3.2.3.1.4. F3: Corte izquierdo de abajo hacia arriba:

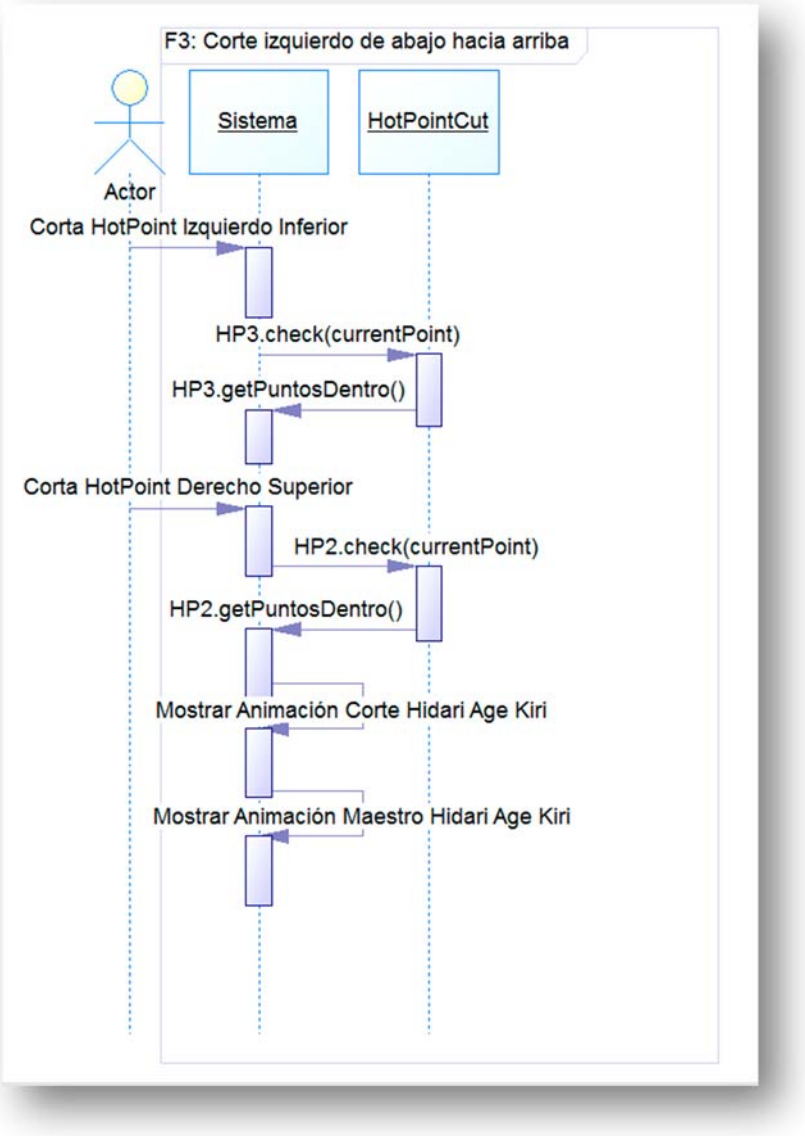

Figura 3-: Diagrama de Secuencia corte izquierdo de abajo hacia arriba.
Elaborado por: David Zurita.

3.2.3.1.5. F4: Corte derecho de abajo hacia arriba:

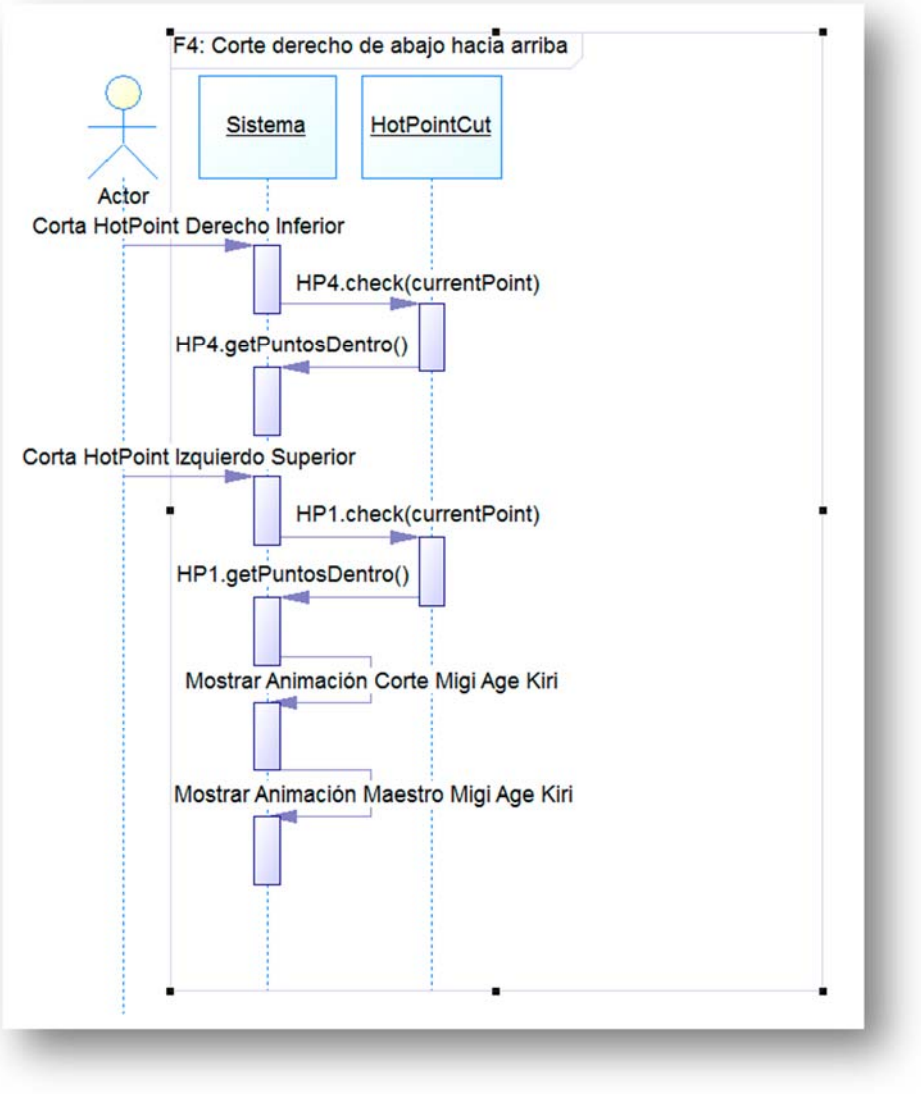

Figura 3-: Diagrama de Secuencia corte derecho de abajo hacia arriba.
Elaborado por: David Zurita.

3.2.3.1.6. F5: Corte horizontal:

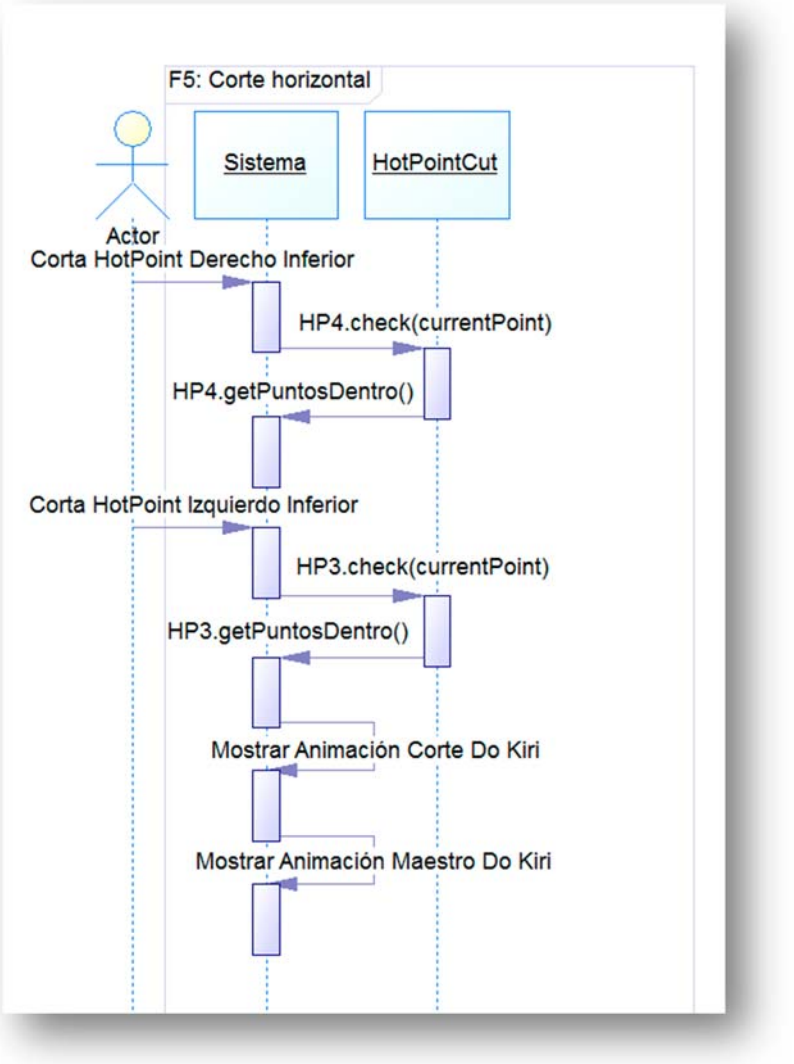

Figura 3-: Diagrama de Secuencia corte horizontal.
Elaborado por: David Zurita.

3.2.3.1.7. F6: Corte vertical:

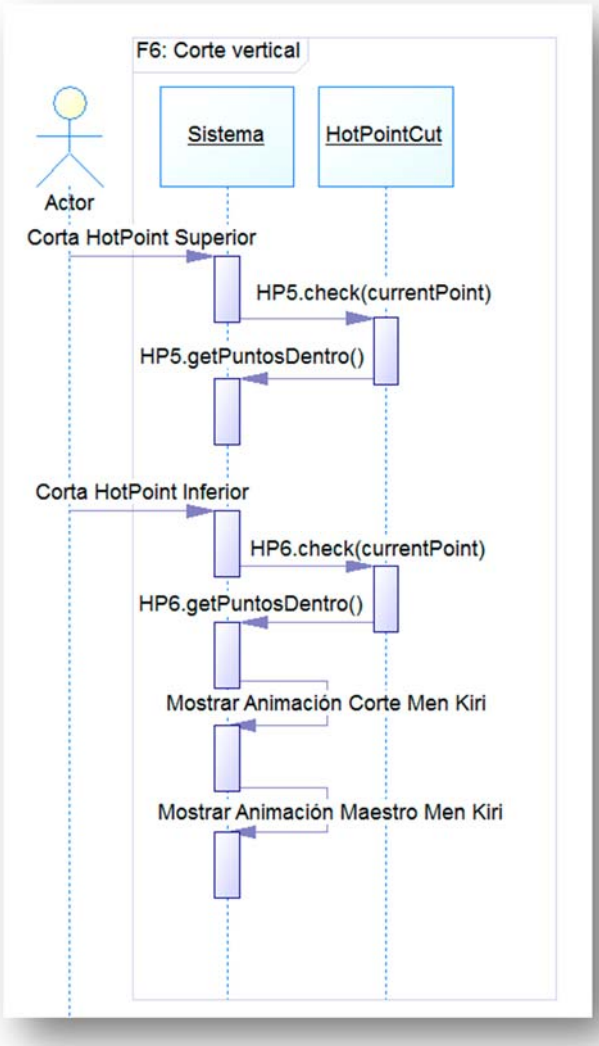

Figura 3-: Diagrama de Secuencia corte vertical.
Elaborado por: David Zurita.

Capítulo IV Implementación

En este capítulo se explica cómo fue el desarrollo dentro de la aplicación, como se expuso en los requerimientos se procederá a realizar un ciclo por cada requerimiento, cada ciclo poseerá una descripción de su realización del desarrollo y la respuesta obtenida dentro de la aplicación, así se podrá observar como la aplicación fue creciendo por cada ciclo que se fue realizando.

Las pruebas especificarán los resultados deseados y los resultados reales una vez ya terminados todos los ciclos del desarrollo.

4.1. Desarrollo.

4.1.1. Ciclo I: Nube de Puntos.

Como se ha venido hablando sobre el espacio que la cámara Kinect reconoce, ahora se debe lograr comprender como esta funciona ya con la codificación, dentro de este ciclo se realiza la programación para la activación del proyecto infrarrojo y el sensor de profundidad para capturar una imagen en la nube de puntos.

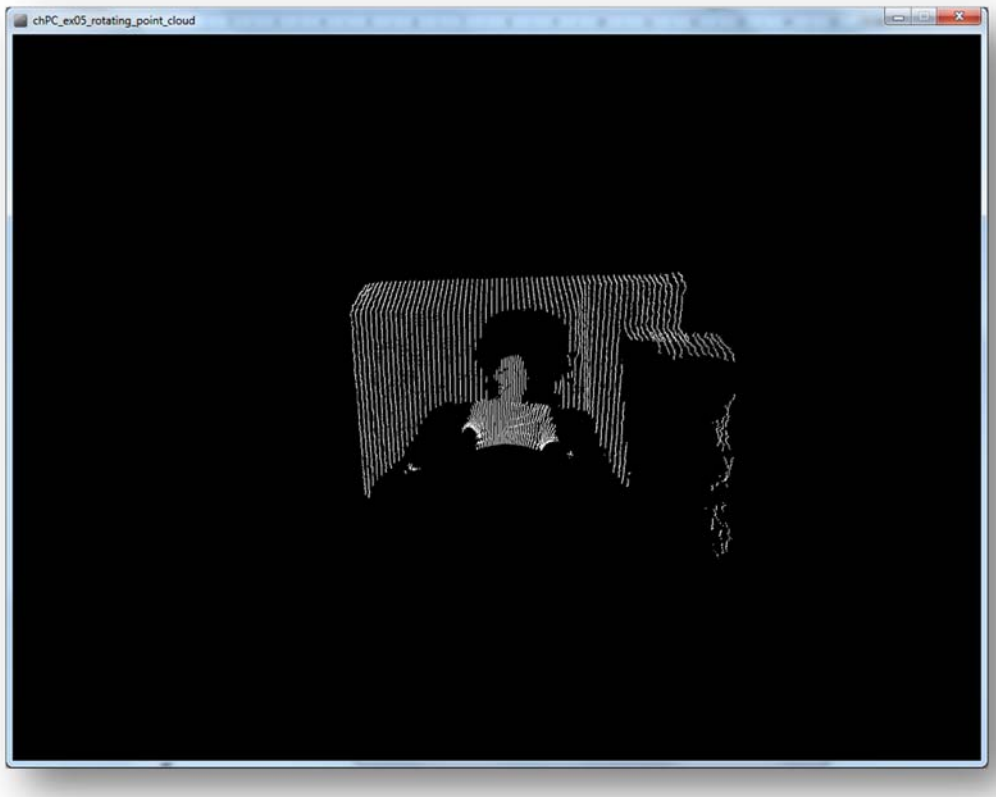

Figura 4-: Nube de Puntos de frente
Elaborado por: David Zurita

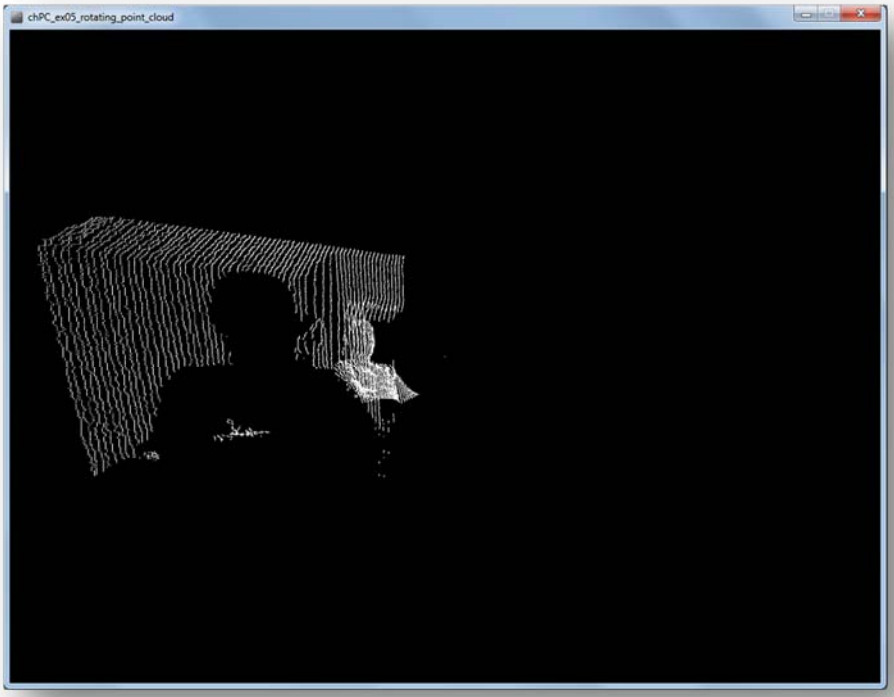

Figura 4-: Nube de Puntos de lado
Elaborado por: David Zurita

Como se puede observar en las dos figuras anteriores, ya no solo se tiene la profundidad de los objetos frente a la cámara Kinect, sino que ahora se puede ver con mucho más detalle las formas, esto se debe a que la nube de puntos que se recibe del proyector es interpretada en tres dimensiones, y con esto se puede ver no solo la profundidad de los objetos sino las formas de estos.

4.1.2. Ciclo II: HotPoint.

Un HotPoint es un espacio determinado dentro de las tres dimensiones, este espacio cumplirá con un objetivo, se puede hacer la comparación con un botón en lo que se conoce hoy en día, la idea principal es ubicar este HotPoint en un sitio determinado, este validará si algo pasa dentro de él y ejecutará un comando si algo pasa, es un botón, pero en las tres dimensiones.

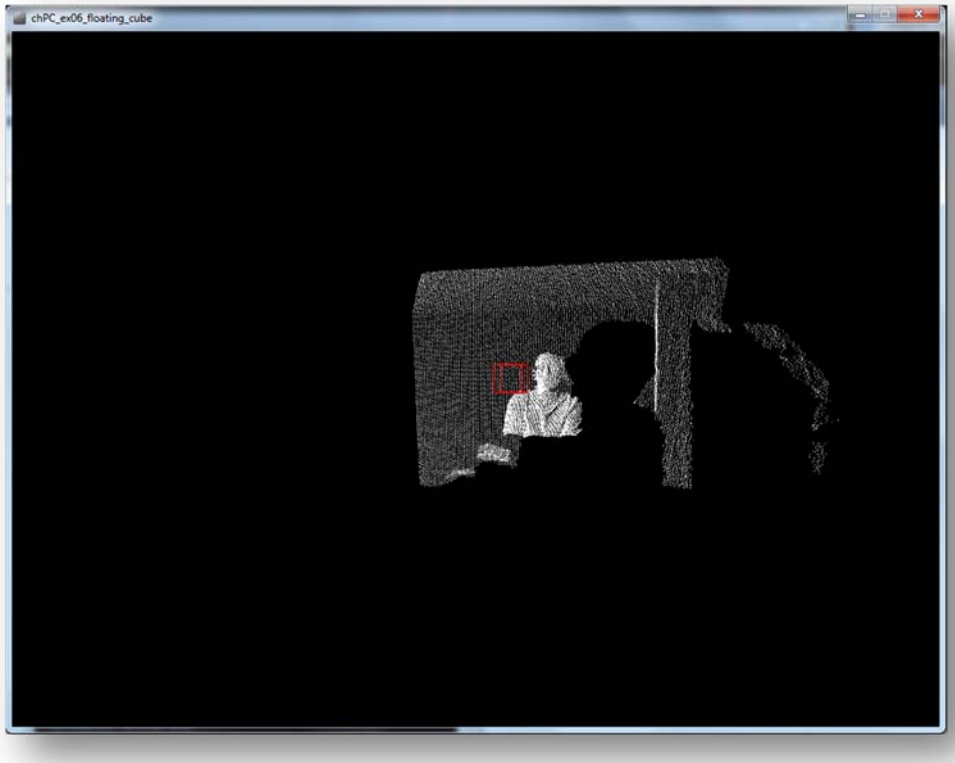

Figura 4-: HotPoint en nube de puntos de frente
Elaborado por: David Zurita

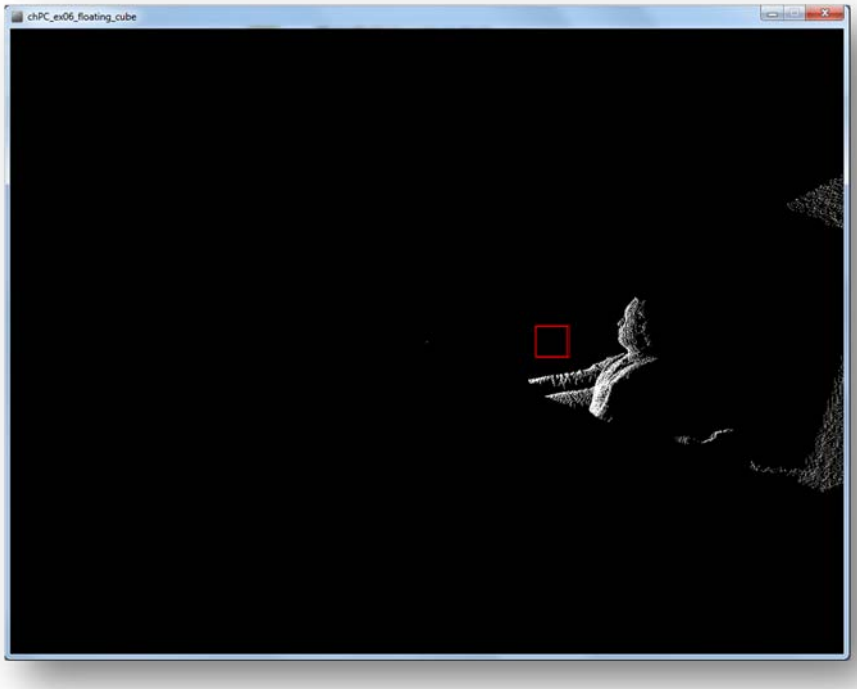

Figura 4-: HotPoint en nube de puntos de lado
Elaborado por: David Zurita

Como se puede apreciar, el cubo representa al HotPoint, este está ubicado en la mitad de la pantalla, se puede ver cómo va girando con la nube de puntos.

4.1.3. Ciclo III: Activación del HotPoint.

Ya que se tiene el HotPoint, este debe poder ser activado al realizar una acción, estas son las bases de la aplicación, se realizarán que el cubo cambie de color cuando algo ha pasado por el espacio dentro de él.

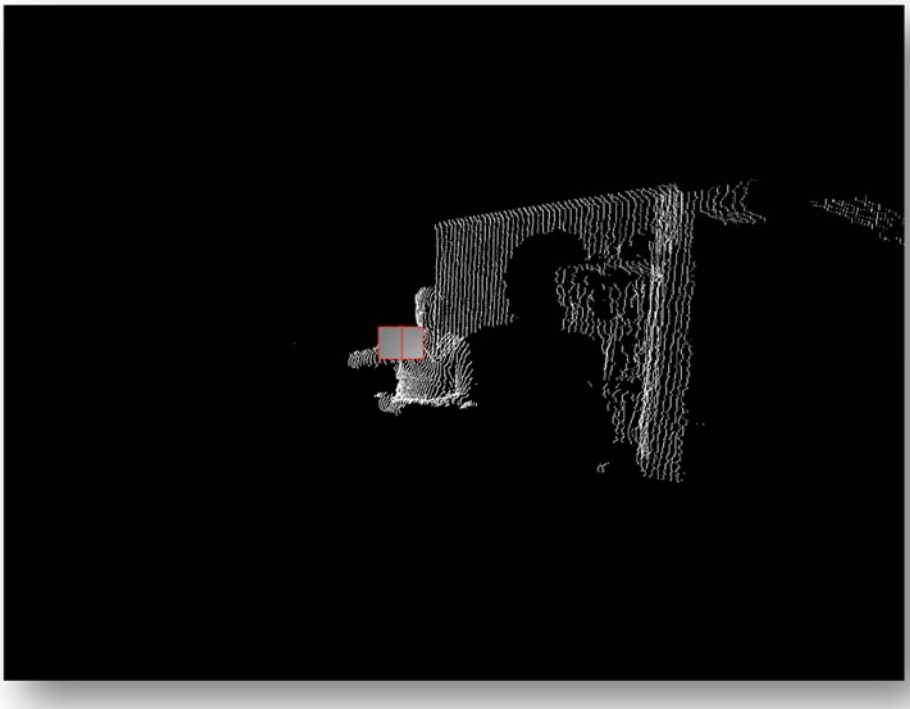

Figura 4-: HotPoint sin activarse
Elaborado por: David Zurita

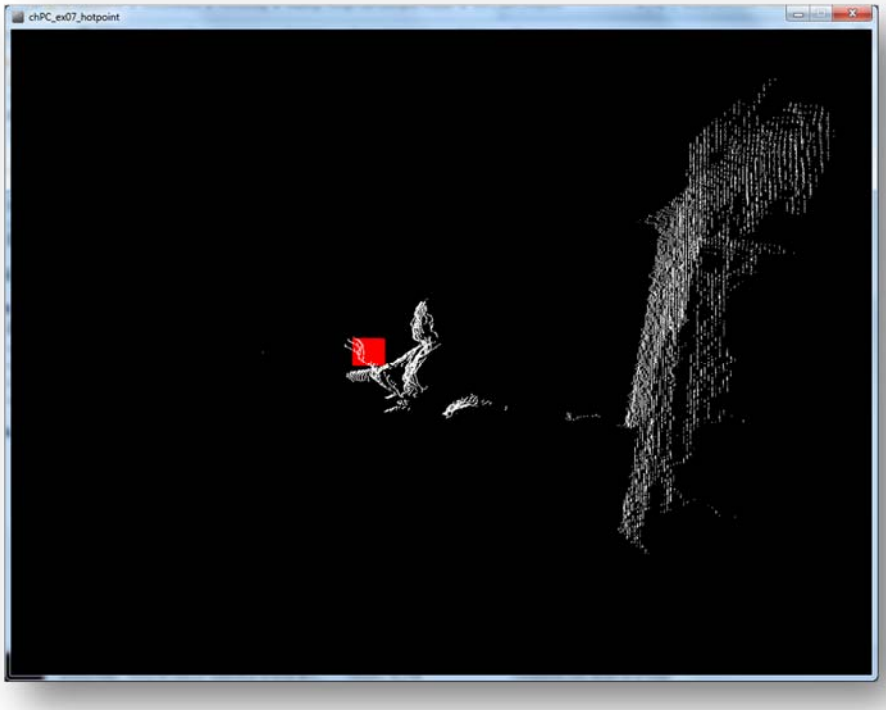

Figura 4-: HotPoint activo
Elaborado por: David Zurita

Como se puede apreciar en la figura 4-6, el cubo se encuentra como gris, esto refleja que no tiene nada dentro, está vacío, pero al colocar la mano, el cubo toma un color rojo, indicando que este se encuentra con algo en su interior, así se realiza la validación del HotPoint, en donde el área asignada no contiene nada se analiza por la distancia, pero cuando en esta área existe algún cambio se lo analiza como si algo estuviera dentro de este espacio, y para visualizar hemos pintado el cubo.

4.1.4. Ciclo IV: Cortes con espada.

Como se ha analizado durante todo el proyecto, el juego reconoce seis tipos de cortes, pero para las animaciones se han producido cuatro animaciones, ya que los cortes diagonales tanto de izquierda hacia derecha y viceversa, son básicamente lo mismo visualmente, por esto se los ha compactado solo en cuatro animaciones por corte, las cuales se puede observar a continuación.

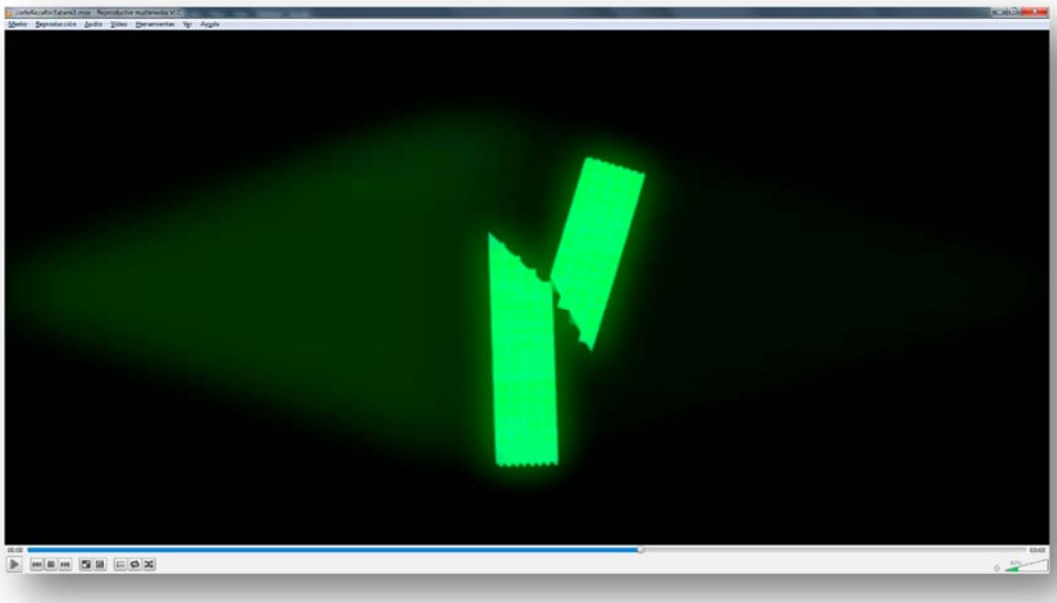

Figura 4-: Corte diagonal izquierdo
Elaborado por: David Zurita

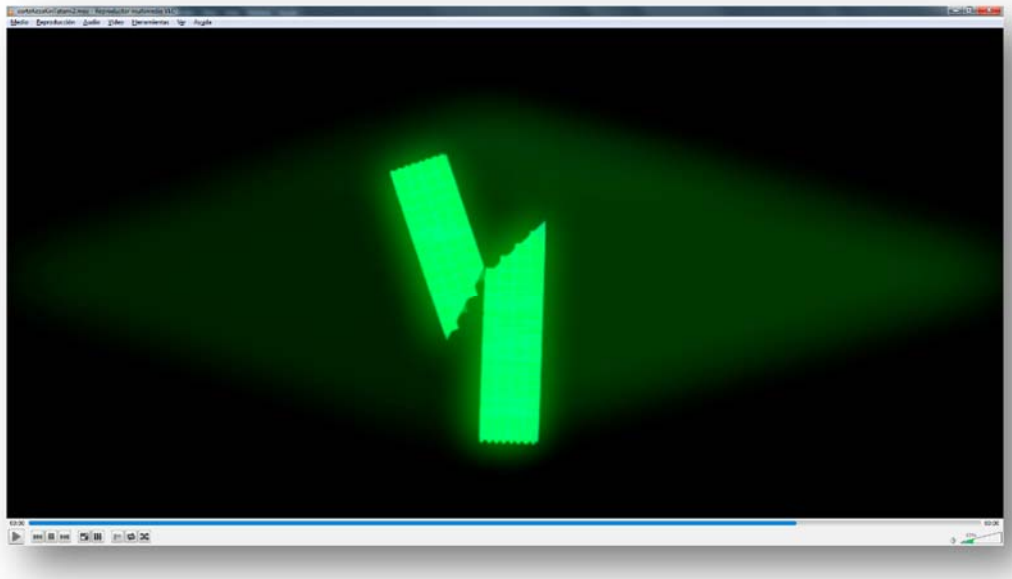

Figura 4-: Corte diagonal derecho
Elaborado por: David Zurita

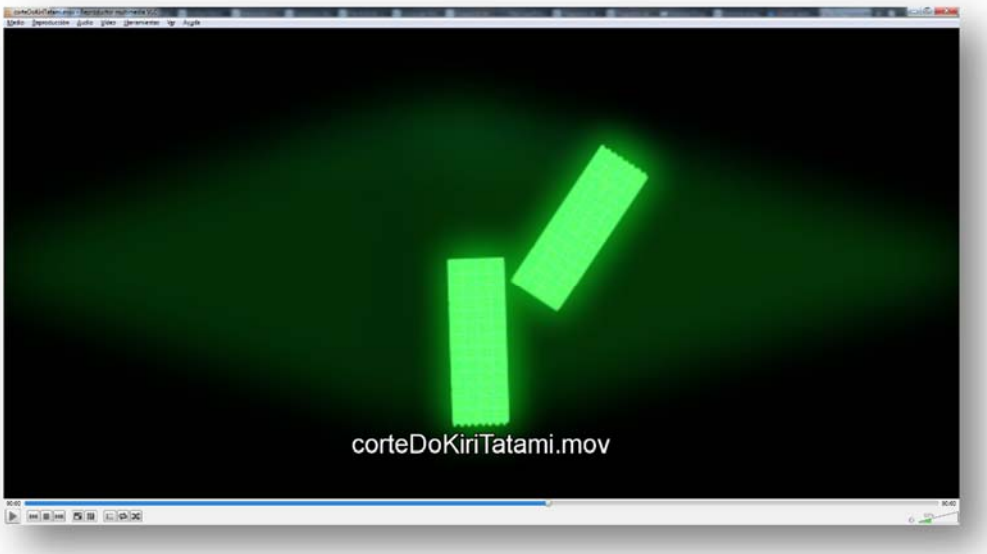

Figura 4-: Corte horizontal
Elaborado por: David Zurita

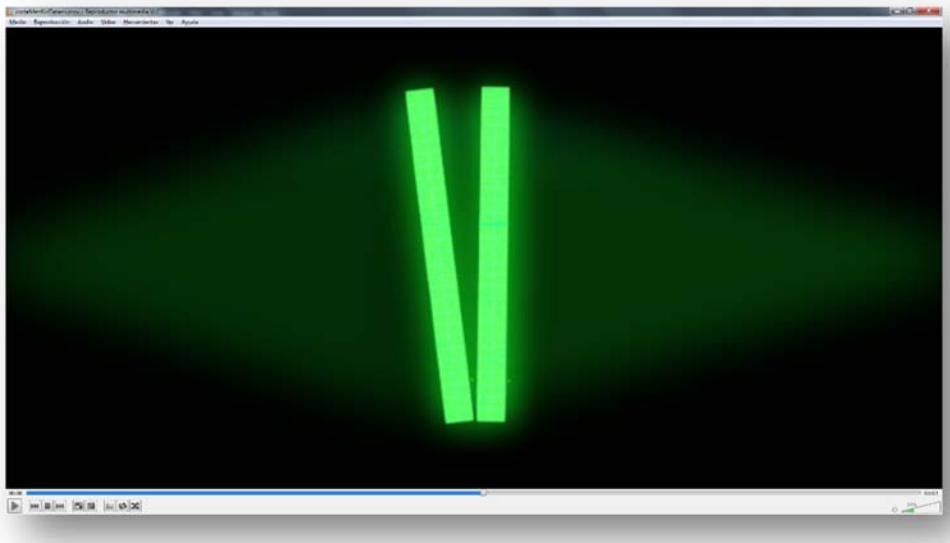

Figura 4-: Corte vertical
Elaborado por: David Zurita

Estos son las cuatro animaciones que se cargan dentro del juego y son las que se pueden observar como respuesta luego de la acción del corte.

4.1.5. *Ciclo* V: Ubicaciones de HotPoints para cortes

Una vez obtenido las activaciones de HotPoints ahora podemos recrear estos HotPoints dentro del juego para que estos evalúen los cortes que ya se ha especificado.

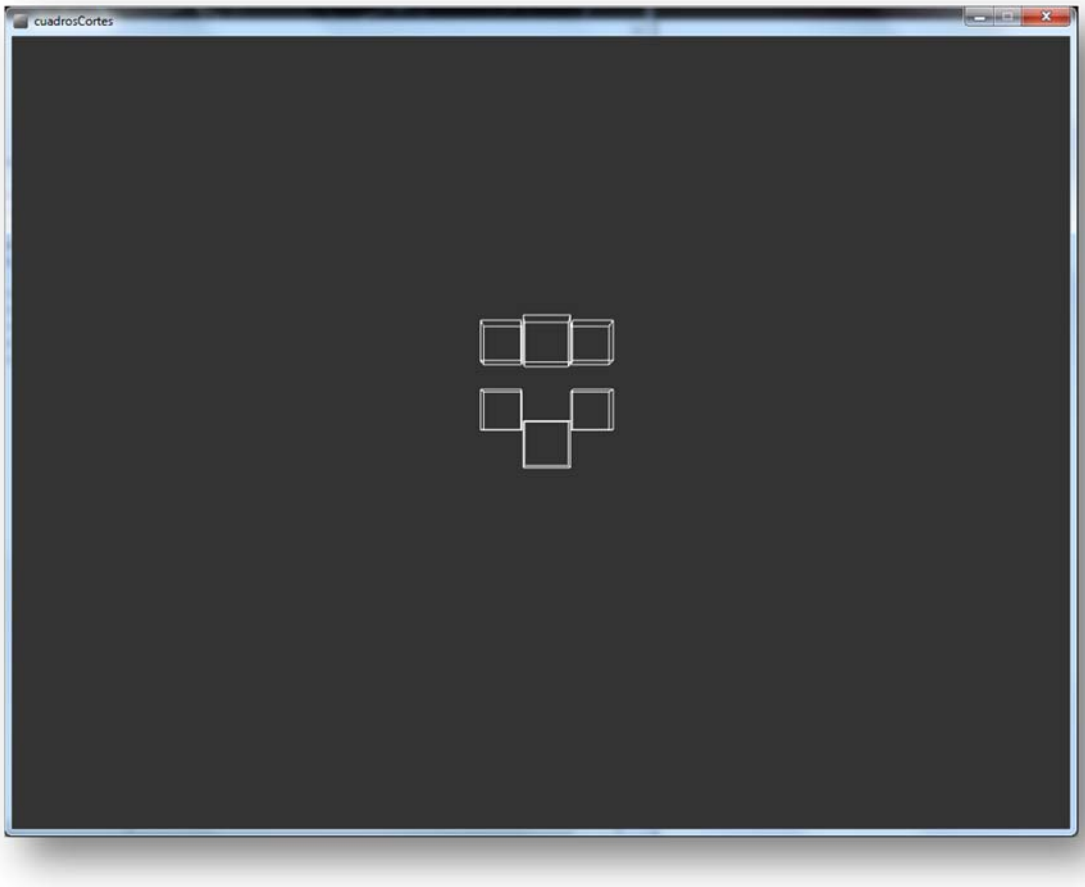

Figura 4-: HotPoints dentro del juego
Elaborado por: David Zurita

La figura 4-11 muestra el espacio que ocupan los HotPoints dentro del juego, estos están centrados ya que ahí ira la parte gráfica en donde se realizará el corte.

4.1.6. Ciclo VI: Integración gráfica.

Ahora todo se debe poner junto, para que el juego tenga algo visual y no solo sea un plano gris con unos cubos que el usuario no va a entender, la parte gráfica fue realizada por un diseñador con la idea de simular un sitio de entrenamiento de artes marciales, pero ponerle su toque especial dentro de un juego. Este es el resultado.

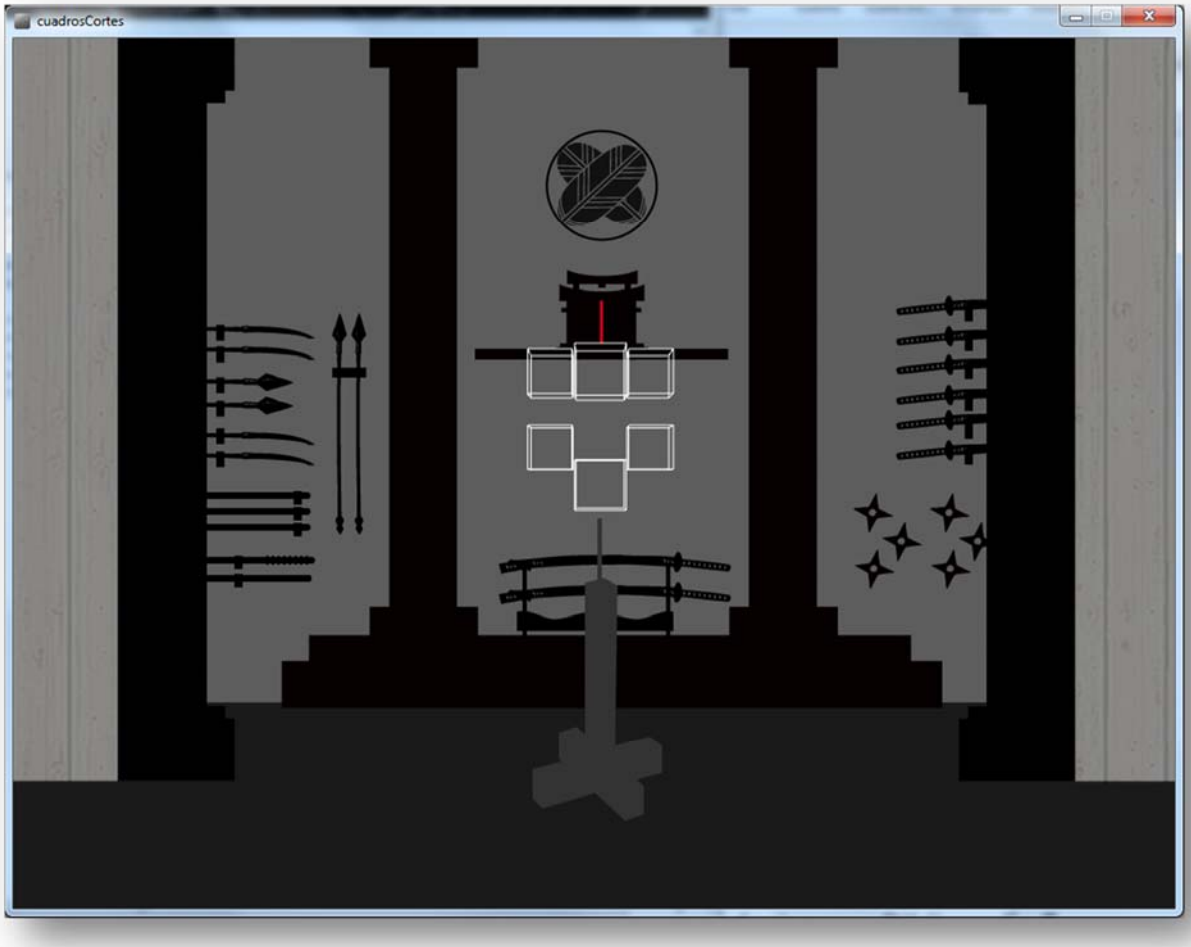

Figura 4-: HotPoints con el fondo del juego
Elaborado por: David Zurita

El fondo que se puede observar es completamente estático, este no tiene ninguna animación, la integración con animaciones será explicada en la siguiente parte

4.1.7. Ciclo VII: *Unificación* con animaciones

La animaciones también fueron desarrolladas por el diseñador, para presentar de tal manera que luzca como un juego, por esto se creó un maestro que indicará al jugador el corte que realice, así como el pedazo a ser cortado, o tatami, parece ser un holograma, que al ser cortado desaparece y se vuelve a construir.

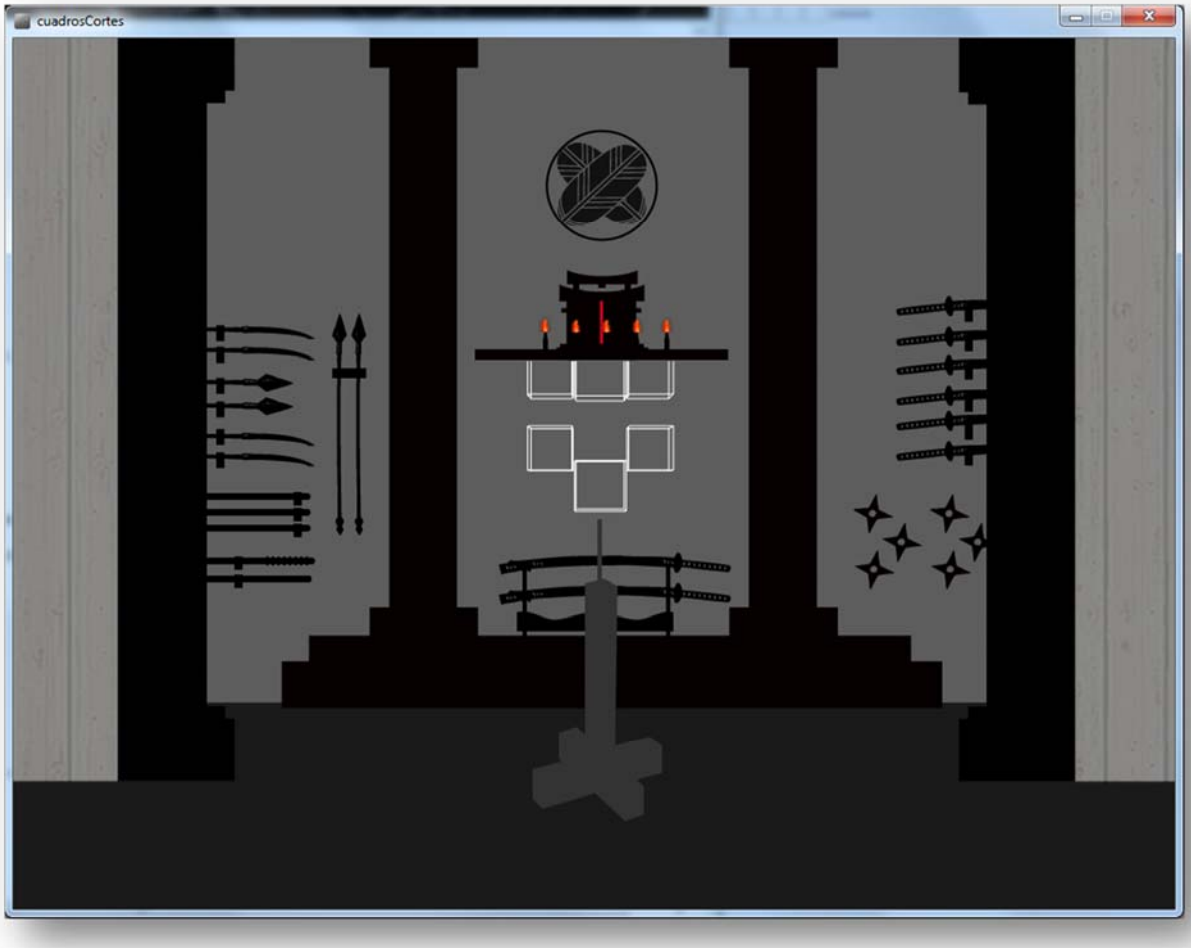

Figura 4-: Velas animadas en la gráfica
Elaborado por: David Zurita

En la figura 4-13 se puede observar que se han colocado velas para dar la visualización de que el fondo no es totalmente quiero y que existe movimiento.

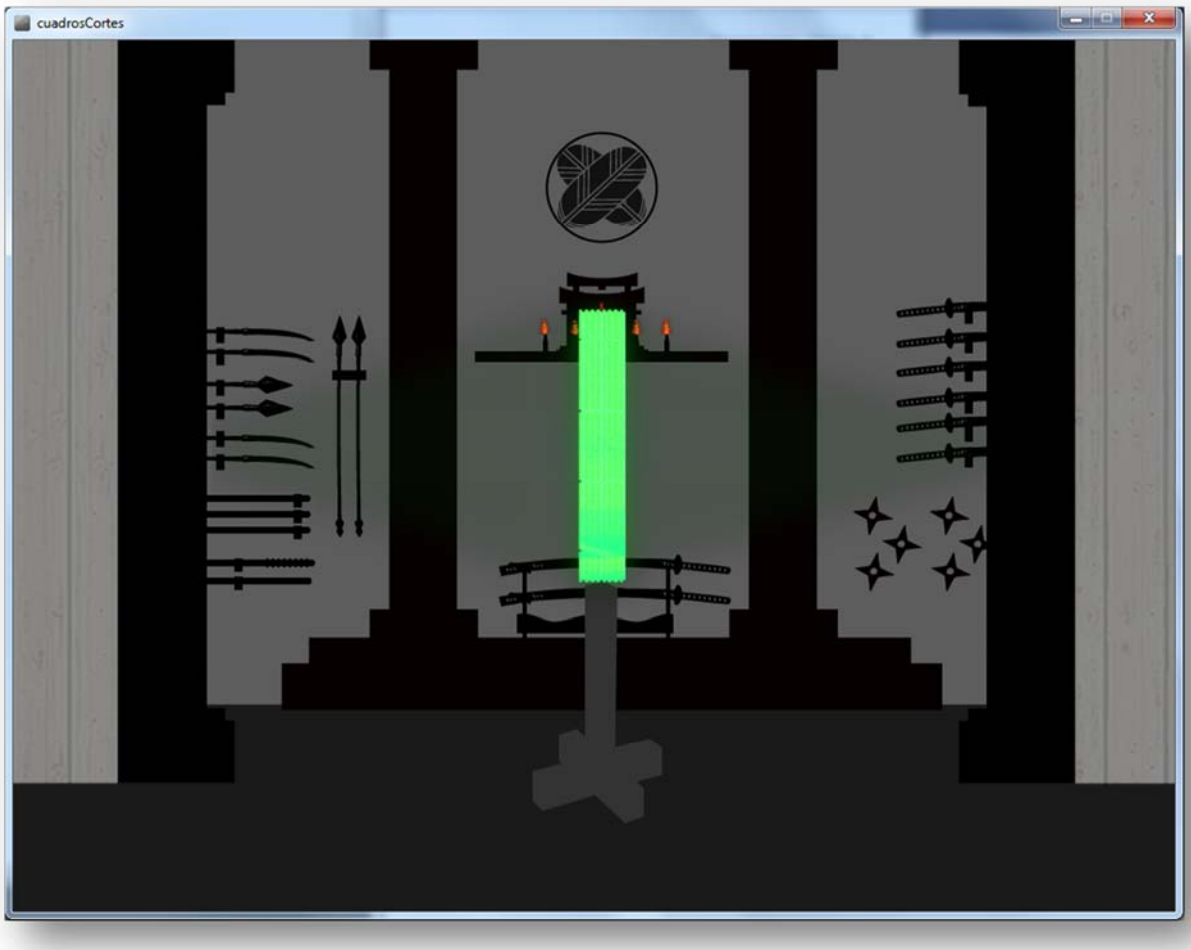

Figura 4-: Ubicación del tatami.
Fuente: Captura de pantalla
Elaborado por: David Zurita

Se ha colocado el tatami, lo que va a ser cortado por el jugador, este al ser ubicado en la parte más exterior del juego cubre los HotPoints creados anteriormente, con esto el jugador no es capaz de observar los HotPoints y se le presenta algo mucho más fácil de entender.

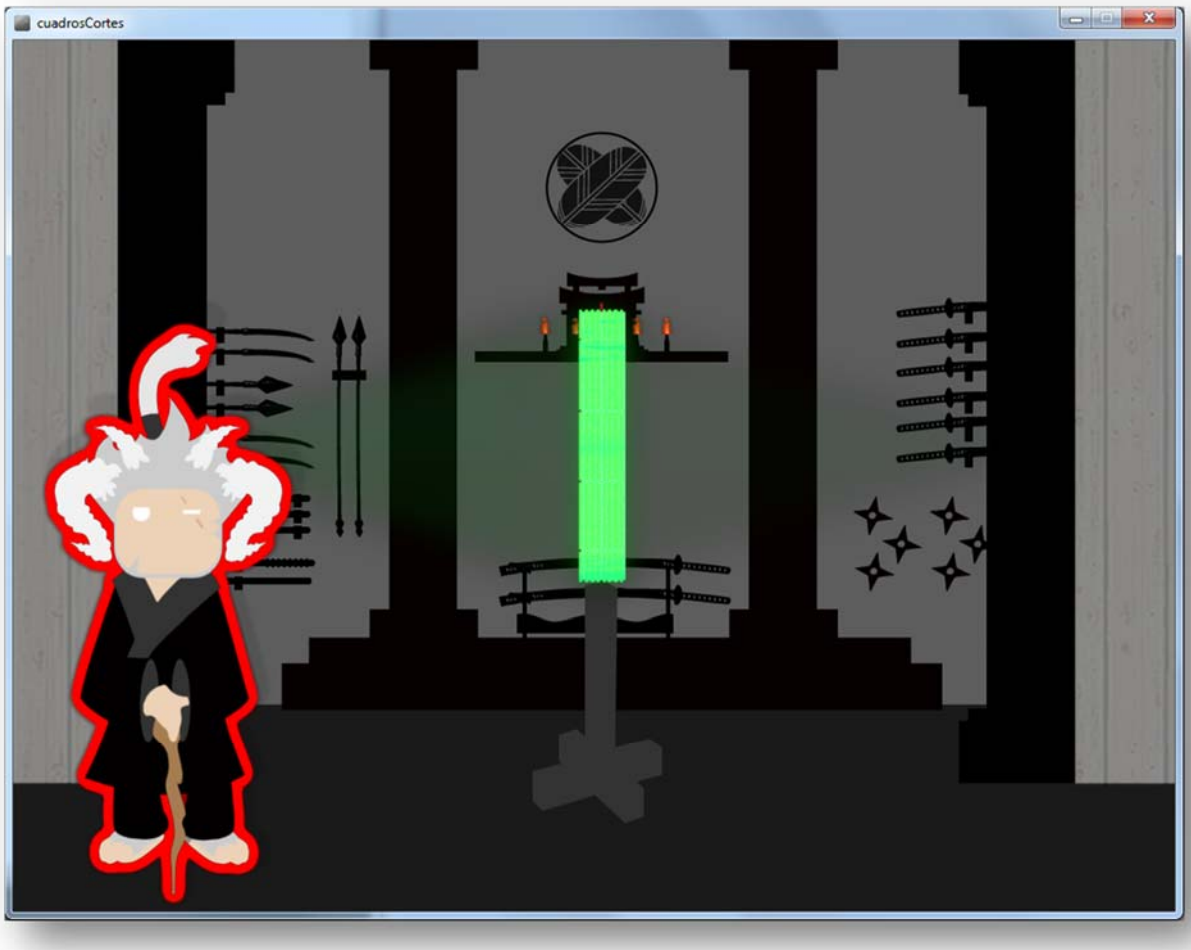

Figura 4-: Ubicación maestro.
Fuente: Captura de pantalla
Elaborado por: David Zurita

Por último se colocó el maestro en la parte izquierda de la pantalla, sobre el resto de objetos, este irá diciendo los cortes que se vayan realizando mientras el jugador realice los cortes.

Capítulo V Conclusiones y Recomendaciones

El último capítulo de la presente disertación presenta las conclusiones obtenidas al desarrollo del juego, así como las recomendaciones que se utilizó para el desarrollo de esta disertación.

5.1. Conclusiones

1. La cámara de profundidad que viene integrada con el Microsoft Kinect presenta un sin número de nuevas alternativas para desarrolladores que deseen crear aplicaciones, donde se pierda el uso de los periféricos que se conoce hoy en día y que los gestos de usuario sean el control de los comandos.
2. Los HotPoints, dentro de las aplicaciones creadas con Kinect juegan una función muy importante ya que al determinar espacios en el campo de visión de la cámara se puede sacar provecho y realizar acciones solo con gestos del usuario.
3. La pequeña curva de aprendizaje que contiene Processing, más su facilidad de crear objetos visuales, da la posibilidad de crear aplicaciones de un nivel totalmente diferente, ya que la programación visual es un campo poco explotado.
4. La integración de Kinect con Processing es verdaderamente sencilla y en muy poco tiempo se puede empezar a desarrollar con esta herramienta; lo que acorta los tiempos de desarrollo y permite la creación de nuevas y complejas ideas.
5. Ya que estas tecnológicas son relativamente nuevas, se tiene un mundo por explorar en las posibilidades que estas brindan; así mismo dado que tienen poco tiempo en el mercado, pueden existir falencias en estas, dificultando el desarrollo de nuevas aplicaciones.
6. Debido a la facilidad visual que presenta Processing, ha logrado simplificar los tiempos de desarrollo de la presente disertación de grado.
7. La unión de comunidades interesadas en ciertos aspectos de la tecnología, llega a crear nuevas y mejoras en la tecnología, como pudimos ver que al crear un premio y la intención de abrir un nuevo hardware para nuevos desarrolladores.
8. La idea de llevar las enseñanzas de un dojo sobre cortes de espada a un programa, se vieron facilitadas por la sencillez que presentan las herramientas seleccionadas para la presente disertación.

9. La abstracción de presentar un dojo virtual, facilita la presentación de estas enseñanzas antiguas, con la intención de facilitar la enseñanza.
10. Ya que se ha usado Processing para la creación del juego, este puede ser ejecutado tanto en Linux como en Windows y Mac, y este no tendrá que cambiar nada en su codificación.

5.2. Recomendaciones

1. Se debe explorar más el campo de las cámaras de profundidad, ya que estas brindan una nueva aproximación a los periféricos para el uso de las computadoras y dispositivos electrónicos.
2. Ya que Microsoft ha liberado su propio SDK para la creación de aplicaciones con el Kinect se puede aprovechar al máximo la herramienta, siempre y cuando esta aplicación sea creada para Windows.
3. El continuar con la investigación sobre el desarrollo de herramientas libres para el Kinect da la posibilidad de llevar estas aplicaciones a nuevas fronteras y comunidades.
4. Processing al ser simple tanto de entender como en su desarrollo, se presenta como una herramienta perfecta para enseñar, tanto las bases de la programación, como lo más complejo que puede realizarse con matemática y paradigmas de programación.
5. Ya que ser un lenguaje más visual, Processing, presenta la capacidad de mostrar resultados sensoriales de una manera más sencilla que otros lenguajes, lo cual es perfecto para ver resultados en poco tiempo e incentivar al estudiante en el estudio de este lenguaje.
6. Proyectos simples de visualización con integraciones de otras tecnologías, logran crear nuevas aplicaciones y capturar la iniciativa tanto del desarrollador como de las personas que usan las aplicaciones, ya que consigo traen creatividad.
7. Este proyecto se lo puede expandir con el uso del esqueleto para el usuario para una mejor captura de movimientos, y cortes más precisos.
8. Se puede crear más módulos dentro del juego para un reconocimiento de armas, con sus respectivos cortes.
9. Se posee la oportunidad de crear una mecánica de puntuación para entender mejor la realización de los diferentes cortes, para perfeccionar estos.
10. Se puede crear comunidades dentro de la universidad, que estas busquen un fin en común, ya que al atacar problemas con gente interesada en un tema, este se vuelve más sencillo y se logran mejores respuestas.

Bibliografía

Digital:

- "Radiación Infrarroja", Wikipedia Org., última vez modificado 28 Octubre 2013, http://es.wikipedia.org/wiki/Radiaci%C3%B3n_infrarroja
- Light", Wikipedia Org., última vez modificado 2 Noviembre 2013, http://en.wikipedia.org/wiki/File:EM_spectrum.svg
- "Infrared", Wikipedia Org., última vez modificado 9 Noviembre 2013, http://en.wikipedia.org/wiki/Infrared
- "CMOS", Wikipedia Org. última vez modificado 12 Noviembre 2013, http://en.wikipedia.org/wiki/CMOS
- "Active pixel sensor", Wikipedia Org., última vez modificado 28 Septiembre 2013, http://en.wikipedia.org/wiki/Active_pixel_sensor

Libros:

- GIORIO, Clemente; FASCINARI, Massimo. Kinect in Motion – Audio and Visual Tracking by Example, Reino Unido: Packt Publishing, Abril 2013 1ra edición. 112p.
- BORENSTEIN, Greg. Making Things See: 3D vision with Kinect, Processing, Arduino, and MakerBot. Estados Unidos de Norte America: O'Reilly**,** Febrero 2012, 1ra edición. 440p
- JANA, Abhijit. Kinect for Windows SDK Programming Guide, Birmingham Reino Unido: Packt Publishing, Diciembre 2012 1ra edición. 392p.
- REAS, Casey; FRY, Ben. Getting Started with Processing. Estados Unidos de Norte America: O'Reilly Media, Julio 2010, 1ra edición. 210p.
- CHROMATIC. Extreme Programming Pocket Guide. Estados Unidos de Norte América: O'Reilly Media, Junio 2009, Ebook. 108p.
- REAS, Casey; FRY, Ben; MAEDA, John. Processing: A Programming Handbook for Visual Designers and Artists. Estados Unidos de Norte America: MIT Press, Agosto 2007, 1ra edición. 736p.

ANEXO 1

Glosario de términos técnicos y siglas

1.1. Términos técnicos

- Radiación Infrarroja (IR): tipo de radiación electromagnética y térmica, de mayor longitud de onda que la luz visible, pero menor que la de microondas.
- PrimeSense: empresa Israelí creadora del chip CMOS APS
- OpenNI: NI nace del acrónimo de Natural Iteraction, es una comunidad donde se desarrolla la librería para el uso de Kinect.
- Processing: es un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java, de fácil utilización.
- PApplet: tiene como padre a: java.lang.Object → java.awt.Component → java.awt.Container → java.awt.Panel java.applet.Applet → processing.core.PApplet. Es la clase base para todas las aplicaciones realizadas con Processing.
- Sketch: así se conoce a las aplicaciones realizadas en Processing, haciendo referencia a un dibujo.
- Dojo: término japonés que significa "lugar del camino". Donde se instruían clases de diferentes artes marciales.
- HotPoint: es un espacio definido, normalmente por una figura geométrica en tres dimensiones, un poliedro, en donde se va a realizar una acción o se va a validar para realizar una acción.

1.2. Siglas

- THz: Terahercios
- nm: nanometros
- GHz: Gigahercios
- µm: micrometros
- APS: sensor de pixeles activos
- CMOS: semiconductor complementario de metal-oxido
- RGB: es un modelo de color aditivo en el que se añaden luz roja, verde y azul juntos en diversas maneras de reproducir una amplia gama de colores. El nombre del modelo viene de las iniciales de los tres colores primarios aditivos, rojo, verde y azul.
- FPS: imágenes por segundo o frames per second
- GPL: General Public License, es la licencia de software libre más utilizada, que garantiza a los usuarios finales (individuos, organizaciones, empresas) las libertades de usar, estudiar, compartir (copiar), y modificar el software. El software que permite que estos derechos se llama software libre.
- LGPL: La Licencia Pública General de GNU o la LGPL (antiguamente la Licencia Pública General GNU Library) es una licencia de software libre publicado por la Free Software Foundation (FSF). La LGPL permite a los desarrolladores y las empresas a utilizar e integrar software LGPL en su propio software (incluso los propietarios), sin que se requiera (por los términos de un fuerte copyleft) para liberar el código fuente de sus propios programas.
- NITE: parte de la librería que maneja todas las articulaciones del usuario y sus respectivas interpretaciones en código
- MIT: El Instituto de Tecnología de Massachusetts es una universidad privada de investigación en Cambridge, conocido tradicionalmente para la investigación y la educación en las ciencias físicas y la ingeniería, y más recientemente en la biología, la economía, la lingüística y la gestión.
- IDE: entorno de desarrollo integrado, por sus siglas en inglés: integrated development environment).

- UML: es un lenguaje de modelado con el propósito general en el campo de la ingeniería de software de creación de diagramas para las diferentes partes de la ingeniería de software.